A Methodology for Aeroelastic Constraint Analysis in a Conceptual Design Environment

A Thesis
Presented to
The Academic Faculty

by

Peter Wilfried Gaston De Baets

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Aerospace Engineering
Georgia Institute of Technology
March 2004

A Methodology for Aeroelastic Constraint Analysis in a Conceptual Design Environment

Approved by:

Professor Dimitri N. Mavris, Advisor	Dr. Jaroslaw Sobieszczanski-Sobieski
Professor Dewey H. Hodges	Mr. Rudolph N. Yurkovich
Professor Daniel P. Schrage	Dr. Daniella E. Raveh

Date Approved: March 22nd, 2004

ACKNOWLEDGEMENTS

A thesis is a one-person endeavor but not possible without the support of numerous people.

My thesis committee deserves particular attention. Sincere thanks to Dr. Mavris for giving me the opportunity when contacting him out of the blue from Cranfield. He provided me with the support when I needed it the most and allowed me to work with great people. Such as: Dr. Hodges, for his timely e-mail answers and immediate availability in his office; Dr. Raveh, for being a friend and mentor I could always turn to; Dr. Schrage, for his many years of background in aeroelasticity; Dr. Sobieski, who during the course of the last three years provided valuable insights and solutions when I was stuck in the details; Mr. Yurkovich, valued for his pointed questions, constructive criticism, and encouragement throughout graduate school. It was a privilege to work in this environment. I am also grateful for the many comments I received on the manuscript and research as it evolved.

My many colleagues at the Aerospace Systems Design Laboratory have helped tremendously over the years. They listened to my computer problems and provided valuable suggestions on this research. I especially thank Jack Zentner, Janel Nixon, Rob McDonald, Nick Borer, Brian German, Peter Hollingsworth, Andrew Frits, Michelle Kirby, Scott Zink, Reid Thomas, and Mike Buonanno. Also Jesus Jimeno and Jesus Rodriguez for their incessant support. I was lucky to have these colleagues become true friends and a family at times.

Lastly, the people whom were always ready to listen and encourage me: mom, dad, Ellen, grandparents, Barbara, Katelijne, and Gert.

This was a very enriching and challenging part of my life at times, and to all of

you: thank you for being there when the road was “Rocky”.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACSYNT	Aircraft Synthesis.
ADOP	Aeroelastic Design Optimization Program.
AIC	Aerodynamic Influence Coefficients.
ALCCA	Aircraft Life Cycle Cost Analysis.
ANN	Artificial Neural Networks.
API	Application Programmer's Interface.
ASTROS	Automated Structural Optimization System.
BLISS	Bi-Level Integrated System Synthesis.
CA	Contributing Analysis.
CALFUN	Calculation of Flutter speed Using Normal modes.
CFD	Computational Fluid Dynamics.
CO	Collaborative Optimization.
CSD	Computational Structural Dynamics.
CSSO	Concurrent Subspace Optimization.
DOE	Design of Experiments.
DOF	Degree of Freedom.
ELAPS	Equivalent Laminated Plate Solution.
FE	Finite Element.
FLOPS	Flight Optimization System.
GA	Genetic Algorithm.
GSE	Global Sensitivity Equation.
GUI	Graphical User Interface.
HSCT	High Speed Civil Transport.
HSR	High Speed Research.
IDF	Individual Discipline Feasibility.

MCS	Monte Carlo Simulation.
MDA	Multidisciplinary Analysis.
MDF	Multi-Discipline Feasibility.
MDO	Multidisciplinary Design Optimization.
NASP	National Aerospace Plane.
NASTRAN	NASA Structural Analysis.
QSBJ	Quiet Supersonic Business Jet.
RDTE	Research, Design, Testing, and Evaluation.
RS	Response Surface.
SA	Simulated Annealing.
SLP	Sequential Linear Programming.
SOAP	Simple Object Access Protocol.
SQP	Sequential Quadratic Programming.
SSA	System Sensitivity Analysis.
SST	Supersonic Transport.
VORLAX	Generalized Vortex Lattice Code.
XML	Extensible Markup Language.

SUMMARY

From the outset, airplane design has been faced with the same technical difficulties; the two primary phases of flight, the take-off/landing and cruise, have different requirements and ideally require completely different airplanes. Problems also arise with high speed vehicles where the fusion of these two phases is more challenging: the amount of active constraints now includes aeroelasticity, noise, and supersonic aerodynamic requirements. The ability to balance and fine-tune the aerodynamics, propulsion, servoelasticity, and structural interactions in a successful package remains a major design challenge.

The objective of this study is the infusion of aeroelastic constraint knowledge into the design space. The mapping of such aeroelastic information in the conceptual design space has long been a desire of the design community. Even though a number of design and optimization tools have been developed to support aeroelastic analysis and optimization in the flight vehicle design process, the toolbox is far from complete.

The conceptual design phase of an aircraft is a multidisciplinary environment and has the most influence on the future design of the vehicle. However, sufficient results cannot be obtained in a timely enough manner to materially contribute to conceptual design decisions. Furthermore, the natural division of the engineering team into specialty groups is not well supported by the monolithic aerodynamics and structures codes typically used in modern aeroelastic analysis.

The research examines how a decomposition method known as Bi-Level Integrated System Synthesis can be adapted to perform as the aeroelastic design tool. This decomposition method separates the disciplines, enables these to individually run

analyses and optimizations, and represents their results through Response Surface models used by the overall system optimizer. The system optimizer uses the Response Surface models to preserve the coupling between individual disciplines. The study describes a comprehensive solution of the aeroelastic coupled problem cast in this decomposition format, leading to the implementation in an integrated framework. The method is supported by application details to a proof of concept high speed vehicle.

Due to the aeroelastic property's highly sensitive nature, physics-based codes are needed that take into account high-definition geometric characteristics of the vehicle. Finite element and aerodynamic panel method codes are used to represent the two disciplines for the decomposition approach. An aircraft synthesis and sizing code was also added to referee the conflicts that arise between the disciplines, for example, when structural weight must be traded for aerodynamic drag.

In order to reduce the initial aeroelastic problem, which is characterized by large data flows, some assumptions are made concerning the drag polars and the loads on the structure. The drag polars in the synthesis and sizing code are based on the baseline vehicle. If the wing changes significantly from this baseline wing, then the aerodynamic penalty or improvement will not be captured by the synthesis and sizing code. A further simplification is the exclusion of a control surfaces in the structural and aerodynamic models. As a result no trim analysis was performed, and correct load assessment was not possible. Care was taken to ensure that a full implementation does not exceed the method's limitations, and solutions were posed to address these assumptions in later iterations. A higher fidelity aerodynamics code is used to generate drag polars online so as to illustrate the possibilities of the proposed method in conjunction with higher fidelity analyses.

The final step of the method involves a scaling approach of the optimized point design obtained with the decomposition method. The analytically generated iso-flutter

and iso-divergence speed lines are plotted in a thrust loading versus wing loading graph. The impact of changing the constraint value can be assessed dynamically. The method also allows for the capture of the impact of mission conditions such as altitude, speed, and angle of attack on the vehicle's flutter and divergence constraints.

The novelty of the present research lies in four points. First is the use of physics-based tools at the conceptual design phase to calculate the aeroelastic properties. This allows for the generation of aeroelastic constraints that are not simply perturbations around a fixed design. This method has the flexibility to allow significant changes in the concept's geometry. Second is the projection of flutter and speed constraint lines in a thrust loading versus wing loading graph. This is a unique result for the design community. The mapping of such constraints in a designer's familiar format is a valuable tool for fast examination of the design space. Third is the improvement of the aeroelastic assessment given the time allotted. Until recently, because of extensive computational and time requirements, aeroelasticity was only assessed at the preliminary design phase. This research illustrates a scheme whereby, for the first time, aeroelasticity can be assessed at the early design formulation stages. Forth is the robustness assessment and the impact of changing speed, altitude and angle of attack could be immediately verified. In such a way, robust design space could be identified.

The method's application to the proof of concept Quiet Supersonic Business Jet resulted in a delta shaped wing for the supersonic speed regime. A subsonic case resulted in a high aspect ratio wing. By using higher fidelity aerodynamics in the synthesis and sizing code, a cranked arrow shape was conserved although modified from the baseline shape. The scaling approach allowed iso-flutter and divergence lines to be plotted. The main effects of speed, altitude and angle of attack on these iso-lines were also discussed. It was possible to identify regions of the design space that were robust to a change in these three parameters. This allowed the selection of a robust

design point while satisfying aeroelastic constraints.

The Design of Experiments and Response Surface methodology allowed convergence of the system optimization but questions were posed as to the accuracy of quadratic models. Other surrogate modeling techniques deserve attention. Improvements by including more disciplines and more detailed models would not limit the applicability of Bi-Level Integrated System Synthesis and are needed to improve practicality of the results. With a modest increase in computer resources, the method could handle more disciplines.

CHAPTER I

INTRODUCTION

From the beginning, airplane design has been faced with the same technical difficulties; the two primary phases of flight, the take-off/landing and cruise, have different requirements and ideally require completely different airplanes.

1.1 The Design Process and Balancing Disciplines

While continually expanding the envelope in speed, distance and payload over the past hundred years, an optimal compromise has been found in current airplane design. For subsonic vehicles, it is possible to conjoin the two distinctly different phases and achieve a well-balanced airplane that allows to take-off and land in short places, yet has a high efficiency in the cruise segment. After a centennial of flight, the high aspect ratio, cylindrical fuselage commercial airplane design has reached a point where not many gains are left for increased efficiency.

Problems sill arise with transonic and supersonic vehicles where the fusion of these two phases becomes more difficult: the amount of active constraints now also includes sonic boom noise and the supersonic aerodynamic requirements. This results in the need for a different set of airplanes and methods, from an operational and design viewpoint respectively.

In order to design a well-balanced airplane which satisfies all operational requirements, an early and correct assessment of the active constraints is needed. In the past decade, research investigated the practicality of using aeroelasticity to the designers advantage. The feasibility of this was proven under various programs such as active aeroelastic wings and others [98, 133].

Designing aircraft that operate in the transonic/supersonic regions presents a formidable obstacle for another reason. Historic databases represent existing vehicles and implicitly account for active constraints, as well as aeroelasticity. Historically regressed design databases, as can be found in Roskam [95], Torenbeek [117], and Raymer [89] have become obsolete as new technologies and vehicle classes are investigated.

An incentive may exist to switch to different configurations such as oblique wing, no-tail configurations, or blended wing-bodies. For these revolutionary configurations, a physics-based approach must account for these pertinent and realistic constraints. However, because most sophisticated detailed *analysis* codes are too computationally expensive for iterative applications, the conceptual designer is faced with a lack of information, limiting a sound decision-making ability at the conceptual phase [24]. There is a need to develop a framework that allows rapid conceptual design of flight vehicles. This would inspire an entire revolution in vehicle conceptual design by enhancing and enlarging the capabilities to do conceptual design with the objective of evaluating novel vehicle concepts [81].

The ability to balance and fine-tune the aerodynamics, propulsion, servoelasticity, and structural interactions for high speed aircraft in a successful package remains a major design challenge [129]. This design process is a long, iterative and arduous process. On top of the complex disciplinary interactions, the vehicle also needs to satisfy stringent environmental, regulatory, and operational constraints. In order to decrease lead time, all-encompassing design tools are needed, preferably validated with flight and wind tunnel data.

The alternative to the physics-based approach is spiral development. Experimental and flight testing gradually improves on a design with small evolutionary steps. After many iterations through different tests and validation of each evolutionary step, the final product may be revolutionary. This however is a costly and time-consuming

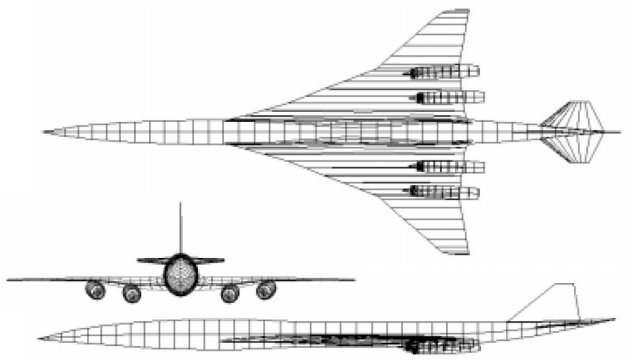

Figure 1: A Notional High-Speed Civil Transport

endeavor.

1.2 Recent High Speed Research

Two decades ago, research was started to address the void of knowledge and tools which aid in making this high speed vehicle design process faster and better. The NASP (National Aerospace Plane) and HSR (High Speed Research) programs had clearly defined goals and milestones. The programs identified key areas where newly developed technology would fit in: CFD (Computational Fluid Dynamics) would help in better predicting critical flight conditions, exotic materials and new manufacturing processes allowed to expand the design envelope, synthetic vision and other technologies enabled the eventual airplane to be cheaper and lighter [123]. Fortunately, computer processing power started raising significantly, and allowed the adoption of more complex analysis codes.

The HSR program evolved into the design of the HSCT (High Speed Civil Transport), notionally depicted in Figure 1. However, as thirty years before with the Concorde, the same requirements drove the problem and eventually led to its demise: only a ten percent mark-up price over current per seat mile revenue was allowed, more stringent Stage III and eventually Stage IV take-off noise, and engine emission constraints. The Concorde, an undisputed technical marvel for its time, was also a

near economic fiasco: excessive take-off noise and sonic boom limited its operational market to coastal airports and over-sea routes [125]. The failure of the HSCT was a logical result since industry and government had, although having established the need for key enabling technologies, invested little in these new technologies to alleviate the old problems.

For the aeroelasticity community, the research programs looked at materials and structural layout. It was symptomatic of most high speed design programs that designers delayed detailed consideration of aeroelasticity until the manufacturing and flight test phase. The principal reasons are in the highly non-linear nature of dynamic aeroelastic characteristics as functions of the governing variables; large volumes of data that need to be generated and interpreted; and long elapsed times required to obtain such data, especially when analysis includes unsteady aerodynamics and FE (Finite Element) models. Optimization that is by its nature iterative amplifies the cost and time implied by all of the above.

The results of these programs for the aeroelastic field concluded with very few new *design* tools. Logically, inclusion of aeroelastic constraints in the aircraft design process remains a challenge despite impressive progress recorded to date (e.g. [12, 15, 22, 28, 52, 116, 133]).

1.3 Research Perspective

The conceptual phase is a multidisciplinary environment and has the most influence on the future design of the vehicle [134]. Design decisions quickly limit designer freedom and increase cost committed as shown in Figure 2 [4]. The ability to prevent costly re-designs and increase upfront knowledge of design decisions, as much information as possible should be made available at the early stage of a design. This paradigm shift is illustrated by Figure 3.

This research will focus on knowledge of aeroelastic constraints. The projection of

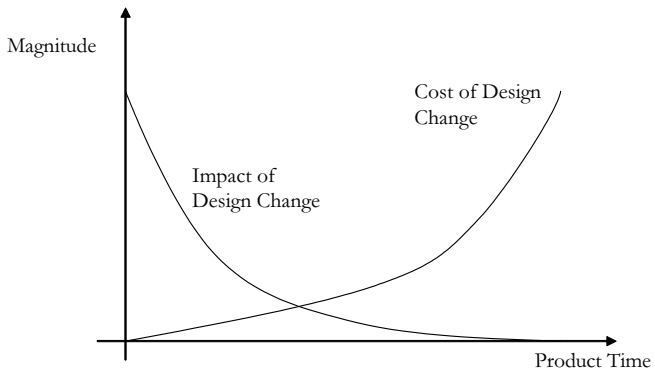

Figure 2: Impact and Cost of Design Changes [110]

such aeroelastic information in the conceptual design space has long been a desire of the design community [5]. This new aeroelastic information should allow the impact of design decisions on aeroelastic characteristics at the conceptual phase to be visualized.

Even though a number of design and optimization tools have been developed to support aeroelastic analysis and optimization in the flight vehicle design process, the toolbox is far from complete. The results often cannot be obtained in a timely enough manner and the natural division of the engineering team into specialty groups is not well supported by the monolithic aerodynamics and structures codes typically found in the above toolbox. In addition, the monolithic codes are not amenable to the use of concurrent processing power available with current computer technology. As optimization is naturally iterative, the process of high-fidelity aeroelastic constraint analysis becomes prohibitively expensive with the current toolbox set. As a result, capturing aeroelastic constraints in a conceptual environment needs to find a solution to two main problems:

1. multiple analysis codes need to communicate efficiently with each other;

2. the obtained aeroelastic information needs to be mapped and visualized in a format conducive to better decision making.

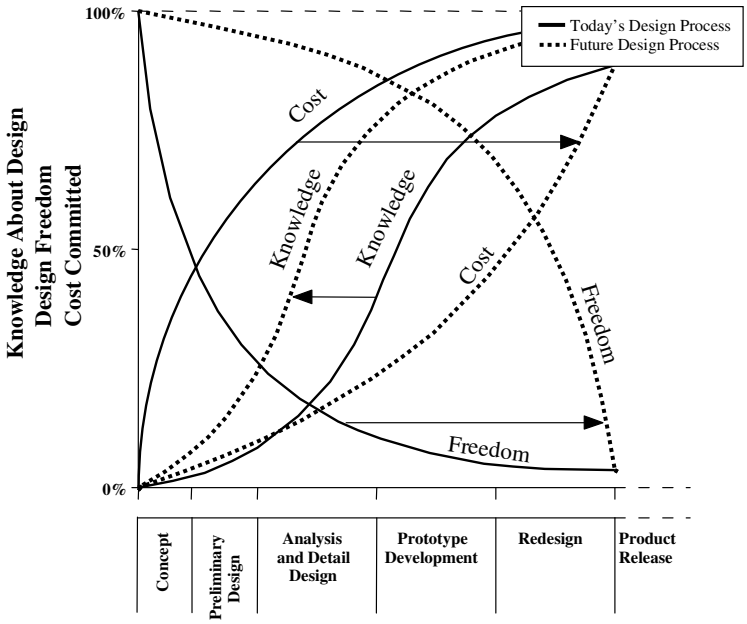

Figure 3: The Paradigm Shift [23]

The research examines how a decomposition method known as BLISS (Bi-Level Integrated System Synthesis) can be adapted to perform as the aeroelastic design tool. The method separates the disciplines, enables them to individually run analyses and optimizations, and represents the results in surrogate models used by the overall system optimizer. The system optimizer uses the Response Surface models to preserve the coupling between individual disciplines. This research identifies and describes a comprehensive solution of the aeroelastic coupled problem cast in this decomposition format, leading to the implementation, and supported by application details to a proof of concept high speed vehicle.

1.4 Introducing the Proof of Concept

With world market projections for approximately one thousand airplanes, interest in high speed commercial airplanes has been re-ignited. While still abiding to the strict environmental regulations and economical expectations, some advantages exist in a

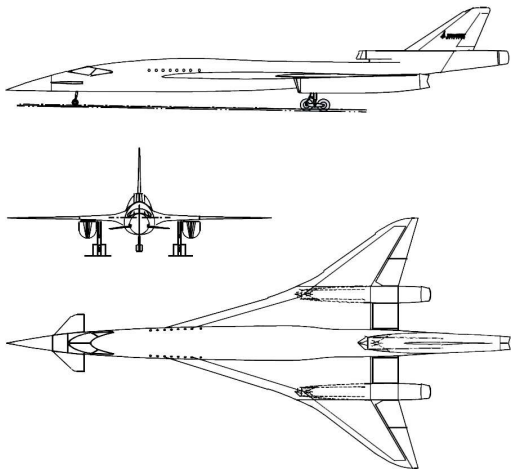

Figure 4: Dassault Aviation SSBJ Concept [60]

6-passenger corporate jet over a 300-passenger airliner such as the HSCT.

- A smaller vehicle reduces the noise concerns to a smaller, more manageable size.
- This vehicle is less sensitive to economic metrics: *"speed is worth the money"* for businesses and the cost-per-seat goal is not applicable to this design [77].
- The quiet aspect allows the airplane to supercruise over land and operate over continents.

1.4.1 Quiet Supersonic Business Jet Market Expectations

Typical characteristics for a QSBJ (Quiet Supersonic Business Jet) were researched. One such a study took the Gulfstream 5 and Global Express characteristics with a range of 6500 nautical miles, allowing non-stop New-York to Tokyo flights in spacious cabins as a starting point [75]. Sixty major worldwide business aircraft operators were interviewed, concluding a QSBJ would be very marketable if the configuration had the following properties:

- Range: carry 6 passengers at least 5000NM with IFR reserves.

- Sonic Boom: low enough to operate unrestricted overland, day or night.
- Field Performance: operate out of 6000ft airfields, preferably shorter.
- Speed: cruise at M1.5 – 1.6 minimum.
- Price: cost no more then $60 million (1999 US dollars).
- Engine Noise: at least Stage III or better on approach and take-off.

Currently, the noise constraint has been upgraded to Stage IV, and the price should be less than $100 million. In addition to the business missions a QSBJ would operate in, there are also military and diplomatic perspectives giving society a faster crisis response capability: aeromedical evacuations, diplomatic missions, transport of distinguished visitors, and special operations (quick reaction) forces deployment.

1.4.2 Configurations Comparison and Design Guidance

Configurations of the supersonic business jet resorted to delta or double delta (cranked arrow) wings. These designs were *evolutionary* from the HSCT and Concorde planforms. Most designs have shied away from variable geometry design for a few reasons. Boeing's Concorde competitor, the seventies ill-fated SST (Supersonic Transport), used a variable geometry configuration depicted in Figure 5. Since the seventies, the F-14, B-1 and F-111 used the variable geometry swing-wing design and what was predictable from a structural point of view was confirmed: the wing pivots and surrounding structure are heavy, expensive to manufacture, and need extensive maintenance attention due to fatigue. Operational success, as the real-life examples prove, is possible but only with a heavy penalty [58]. As such, most military designs have shied away from this configuration and this could be handwriting on the wall. A list of problems with this configuration are [124]:

- Aerodynamic center shift with sweep and Mach number. As the wing centroid is moved aft and the aircraft goes supersonic adding a center of pressure shift,

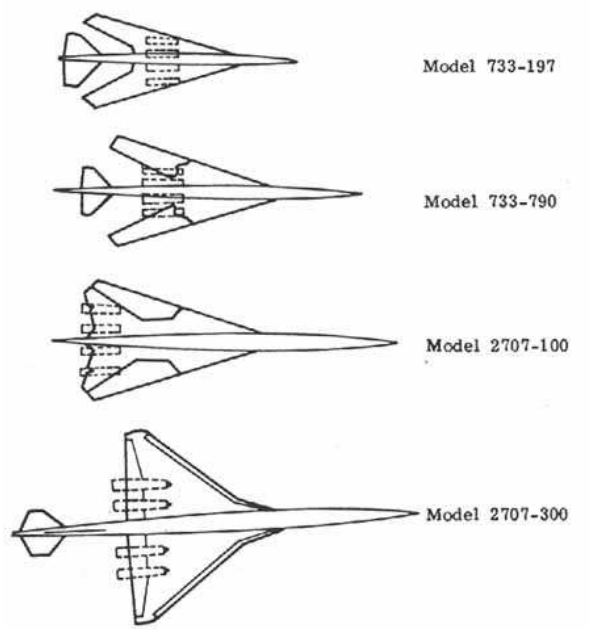

Figure 5: Boeing SST Designs

there is a wide range of the aerodynamic center. This results in heavy trim drag penalties associated with fuselage and tail loads making the aircraft heavier, and possibly complex, costly fuel systems.

- Wing seal and fairing are mechanically complex and add drag because these need to be accommodated into the fuselage design. This increases the wetted area, which adversely affects supercruising.

- Wing carry-through of loads is not as efficient as with fixed wing designs. The pivots are usually separated from the main fuselage to reduce aerodynamic center shift. The required structure to transfer the bending moments to and through the fuselage results in heavy pivots. This volume in the fuselage could otherwise be used for fuel which as a result needs to be accommodate somewhere else in the fuselage.

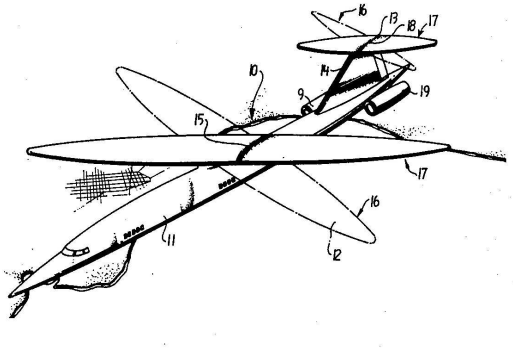

Figure 6: Oblique Wing Supersonic Aircraft [46]

An alternative exists taking advantage of variable geometry, while eliminating some of the drawbacks listed above. The oblique wing concept first appears in the late forties [46, 47]. Oblique wings use a single, centrally located pivot point, producing greater effective geometric change. Research into oblique wing concepts shows high potential to achieve success: sweeping of the entire wing with one central pivot reduces the requirement for a structure to carry the bending moment to the fuselage. For equal low-speed performance, the oblique wing shows reduced wave drag because a higher fineness ratio can be achieved, illustrating the better supersonic qualities of an oblique wing variable geometry [83].

Concerns of divergence initially pushed designers away form this configuration. However, testing quelled the divergence problem. In the oblique wing configuration the airplane is free to roll to alleviate some of the load on the forward wing [58]. The design challenge remains because the rolling tendency must now be controlled, and at least statically the forward-swept wing must be stronger.

Other remaining drawbacks are:

- The pivot mechanism must be fail-safe. Unlike the landing gear, safe operation is required throughout the flight, not simply at taxi, take-off and landing [7].

- The control system architecture of an oblique wing must account for the complex

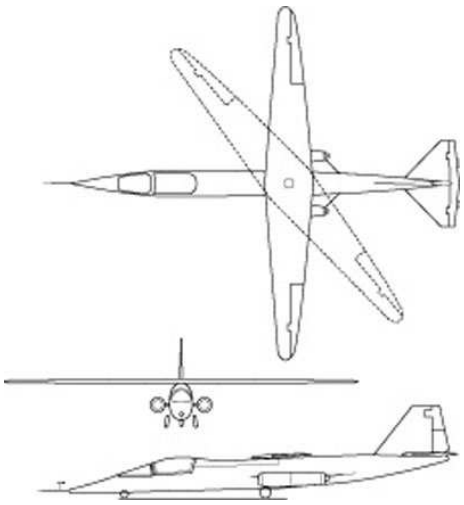

Figure 7: NASA AD-1 Oblique Wing Demonstrator

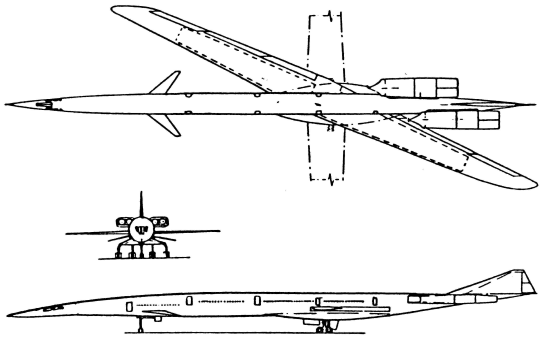

Figure 8: Oblique HSCT Concept [29]

structural dynamics and cross-coupled aircraft response while retaining the feel of a conventional airplane to the crew.

- The certification of the pivot mechanism and control surface system presents a challenge due to the lack of clear regulatory guidelines.

- From a human factors perspective, the pilot and passenger community must be convinced in accepting and trusting an unusual configuration.

Table 1: Oblique Wing Supersonic Transport Features [32]

Features	Assests	Liabilities
Aerodynamic		
Cruise, Supersonic	+L/D	
Cruise, Subsonic	++L/D, +Ride	
Take-Off, Landing	+++L/D, -Weight -Velocity, +Safety	
Aerodynamic Center	Smaller Shift	
Aero Ruling (none)	- Weight, +Plug	
Propulsion (Caravelle mounting)		
Wing (no engines)	Simple Structure	-Bending Relief
Exhaust Noise	Aft Passengers	+ Structural Fatigue
Engine Inoperative	Less Yaw Moment	
Inlet	+Safe, -F.O.D., -Spray	+Distortion
Nozzle Length		-Aircraft Rotation
Turbine Failure	Aft passengers & Fuel	
Landing Gear	-Length, -Weight	
Materials and Structures		
High Lift (T.E. only)	-Weight, -Complex, -Cost	
Fuselage (constant cross-section)	-Weight, -Cost	
Empennage (smaller)	-Weight, -Trim, -Yaw	
Nacelle Attachment		+Weight
Utilization		
Airport	-Spot, +Gate Access	
Access	+Doors, +Safety	
Floor (flat)	-Cost, -Weight	
Variable Geometry	+Safety	
Sonic Boom	-Wave Drag	

A summary of the major design features of an oblique wing concept can be found in Table 1.

CHAPTER II

BACKGROUND

In the previous chapter a motivation and need were described for new tools in the design process for high speed vehicles. In order to describe this new design tool, an overview is needed of sources of information, current design practices, and why is the design process done that way. In order to achieve this, the following questions need to be addressed:

- How do designers take aeroelasticity into account at the early design formulation phases?
- What has to change in the architecture and problem formulation?

2.1 Design for Aeroelasticity during the Early Design Formulation Phases

Essentially, there are three ways to approach design-for-aeroelasticity. The first one is a special case with aeroelasticity being an after-thought. This design practice has vanished in all but the experimental (kit-built) and low speed general aviation airplane design since the discovery and maturing of the aeroelasticity research domain. The second design practice starts from an already optimized or existing geometry and aeroelastic ingenuity is applied to comply with regulations and requirements. Although this approach works well, it may result in a sub-optimal design for revolutionary concepts. The third practice would be a structural optimization concurrently executed while meeting the aeroelastic constraints. This is the best option as far as achieving design optimality.

Most if not all designs perform an aeroelastic optimization of an already optimized flight vehicle, falling in the second category above. This is due to most commercial vehicles (airlines and business jets) being extensions or modifications of already existing geometries, i.e. families of vehicles. Another reason might be that the vehicle to be designed is a conventional configuration (*cookie cutter design*), for which the "eccentric" aeroelastic behavior and problem areas are well-known [57].

As a result, there is a general perception that classical aeroelastic effects (divergence, flutter, aileron reversal etc.) are today mainstream in design, analysis and certification [90, 122]. However, basic information on aeroelasticity and the impact of structural changes on an conventional airplane have largely been based on knowledge gained through experiments and research. For instance, the effect of spar placement, or of sweep on flutter are well documented for conventional planforms.

Despite this practical knowledge, aeroelastic problems or unacceptable aeroelastic behavior are not uncommon for an initial wing design due to a lack of aeroelastic analysis at the conceptual design stage. This aeroelastic problem is then corrected in an *ad hoc* manner, resulting in a sub-optimal design [62]. Furthermore, there are many vehicle concepts where aeroelastic effects are pervasive and new that progress has to be made in the analytical capabilities [122]. Knowledge of these aeroelastic properties and having the ability to assess aeroelasticity early is crucial and becomes a prerequisite to success in the design of future high performance or revolutionary concepts. Komarov and Weisshaar describe the need as the ability to generate *analytic* estimates that are not tied too closely to a historical database as a key need for exploring new design spaces [57].

Accurate and timely predictions of aeroelasticity are a necessity for efficient design of modern aircraft [27]. Dynamic aeroelasticity is the single most important aeroelastic constraint in the design environment because of the difficulty to correctly assess in the early design phases. Zink et al. [134] indicate that Boeing's SST design

and NASA's HSR and HSCT programs left the flutter problem unresolved. This was due the time allotted and resources available not being sufficient to allow the analysis to be performed. Both Boeing's and NASA's research groups concluded that flutter would be determined in the prototype development phase [65]. It is at the early design stage though that the complex problem can be most effectively and economically addressed [134]. According to Weisshaar [122] and Ashley [9] consideration and *prediction* of aeroelastic effects are essential for the efficient design of high performance flight vehicles.

2.2 Process, Tools, and Techniques in Design

An exposition of the structural design process and the differences between a design and analysis tool will highlight voids preventing aeroelastic knowledge from being included in the conceptual design phase. The availability of multidisciplinary optimization techniques is instrumental in finding tools to fill this void.

2.2.1 Structural Design Process

The structural design process is an iterative process between control law design, load analysis, and structural analysis/design of which the major elements of the design process are highlighted in Figure 9 [133]. This process is driven by design requirements and criteria. It is an iterative loop, which starts with the loads model, typically a FE model including mass and stiffness, an aerodynamic grid and sometimes a preliminary flight control model. The next step involves a load selection from a variety of load conditions: different fuel loads, different payloads, different flight and ground maneuvers, fatigue conditions, etc. A few hundred external loading cases are selected after extensive scrutiny from a team and are deemed critical to the structural design.

The requirements for that vehicle dictate certain maneuvers and the vehicle is flown through these g-maneuvers with a flight simulator. The wing root bending

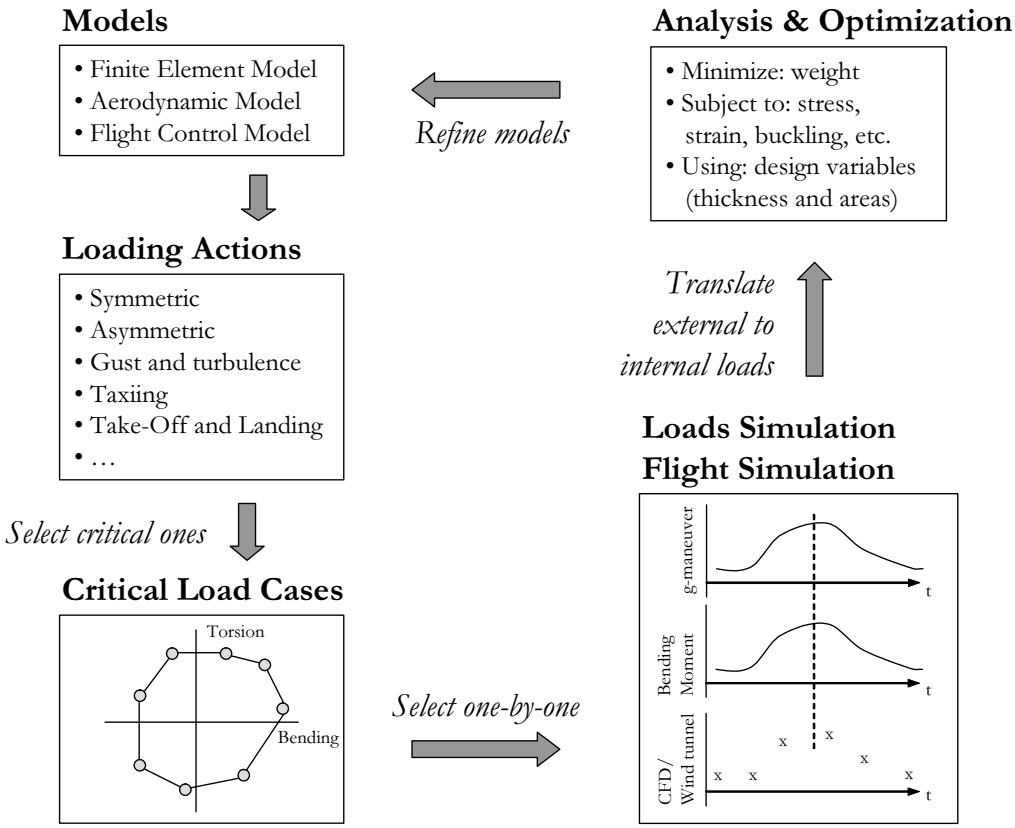

Figure 9: The Notional Industry Design Process

moment, torsion moment, and other external forces are tracked and plotted. Some points are chosen and validated with either wind tunnel or CFD data. Using a trim analysis, the correct loads are then calculated. In the final step, these few cases are translated to the FE model and the structure is then optimized to meet various constraints such as stress, strain and buckling, while minimizing weight.

Iteratively, the loads model is now updated with sizing and mass results from the first iteration. This process is repeated all the way up to testing and evaluation of the eventual airframe until designers are satisfied with the resulting structure, and that it will successfully complete its mission. During this iterative process, the models become of higher fidelity as the design progresses. Since lower fidelity codes are used at the outset of the process, major design changes are often required due to inaccurate

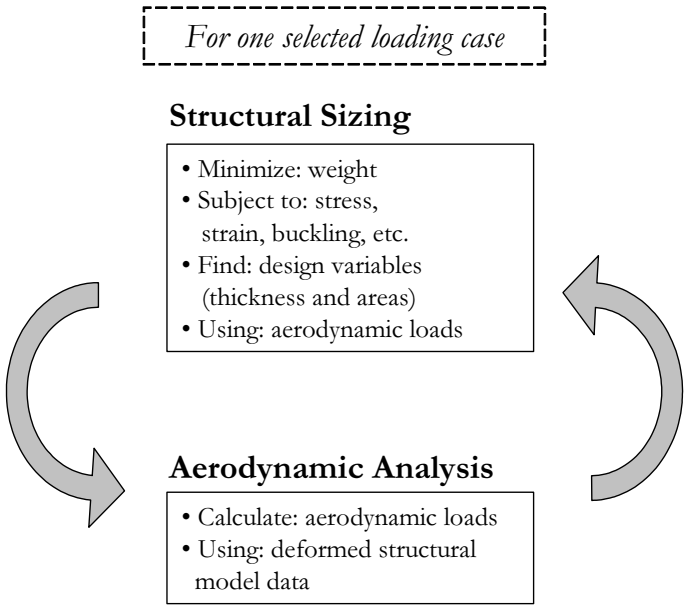

Figure 10: The Notional Academic Design Process

initial load prediction or wrong initial assumptions on critical loading cases.

This real-life example is in stark contrast with the typical load loop used in academic structural optimization [132], as shown in Figure 10. The conceptual design assumptions that air loads are based on linear theory and that the structural weights computed from the FE model are accurate, both lead to erratic structures when compared to industry practices. In both cases, the uncertainty due to initial low-fidelity tools of the industrial design process and the simplified structural design process used, can be solved by either making the structure *robust* to uncertainty in the loads, or else by more accurately predicting the loads at the design process outset. After design feasibility is achieved, the next important aspect is assessing the design robustness as quickly as possible [130].

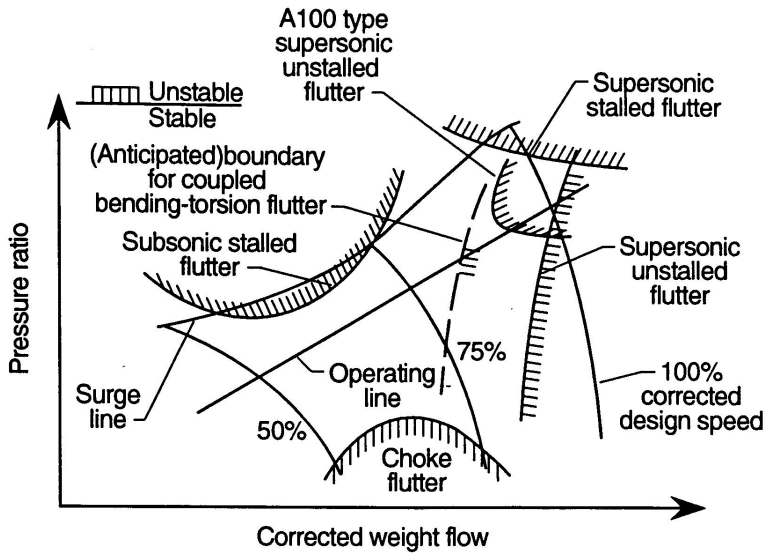

Figure 11: Example of Constraints in a Design Environment for a Compressor [14]

2.2.2 Design versus Analysis Tool

A design tool should add value to the design process by providing technical support for design decisions [110]. Analytical tools create data, which to be useful, needs to be presented to the process as communicable and descriptive information at the right time. Information needs to be mapped to the design space and visualized in a format conducive to making better decisions, taking into account the high-definition geometric characteristics, computational effort, and realistic constraints. The type of information required by a conceptual designer when compared to what a detailed designer expects, is significantly different. This visualization disconnect is what prevents most (analysis) tools from being adopted in the conceptual design phase.

Data representation and a mapping to a thrust loading versus wing loading chart of these constraints is critical for conceptual designers. Examples of such conceptual charts are Figure 11 for a compressor and Figure 12 illustrating this for a fixed-wing fighter airplane.

Apart from visualizing the data, a separate research track has tried to create

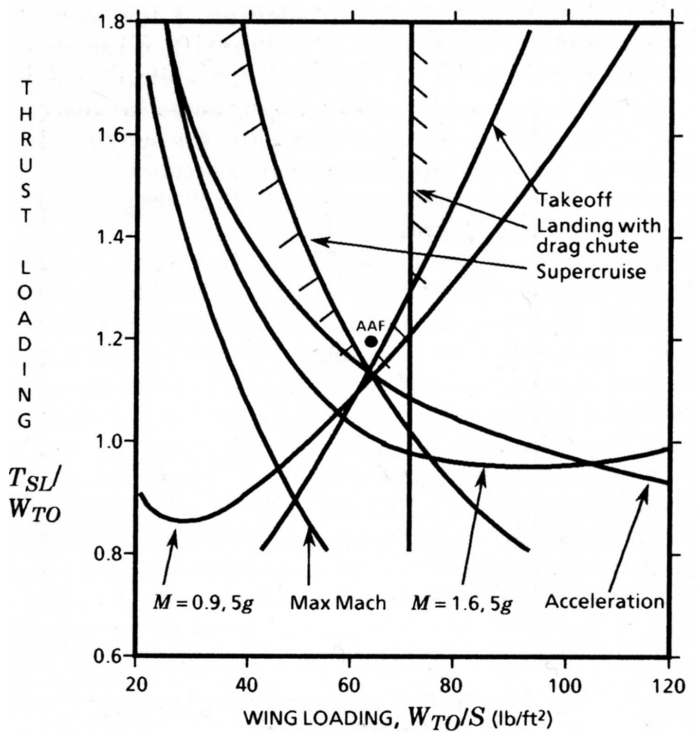

Figure 12: Example of Constraints in a Design Environment for a Fixed-wing Fighter Airplane [66]

design tools that can be used at the conceptual level by using simplified theories and models. Results obtained with these tools are good and have been validated with FE models and experiments. Examples include ELAPS (Equivalent Laminated Plate Solution) [34], CALFUN (Calculation of Flutter speed Using Normal modes) [33], ADOP (Aeroelastic Design Optimization Program) [25], and ASTROS (Automated Structural Optimization System) [82]. Drawbacks of these tools are:

- These remain stand-alone analysis codes, i.e. these are not easily integrated in a design environment.
- These do not provide enough flexibility to swap in higher fidelity codes and models as these become available.
- Their usage requires a good knowledge of applied model simplifications and code assumptions [93, 109, 120, 121].

- Due to simplified models being used, to quickly analyze perturbations or radically different designs with the same code may be difficult.

- Dynamic aeroelasticity requires computationally expensive extraction of eigenvalues from the FE model. This model consists of thousands of DOF (Degree of Freedom). Inclusion of dynamic aeroelasticity is made difficult since the behavior relies on detailed structural information, which the above simplified codes may not provide.

For accurate aeroelastic knowledge, a conceptual FE model is required coupled with an aerodynamics code [94, 130]. The aeroelastic properties require such detailed codes. Mostly due to increased computing power, the execution times for a conceptual model have been decreasing [119]. As a result, such codes can actually be used now in a somewhat iterative fashion. The last element missing is multidisciplinary optimization.

2.2.3 Multidisciplinary Design Optimization Techniques

A clearer picture can be drawn of the state of affairs and why this is the case: the long execution times of the analysis codes, together with the high expectations attached to the aeroelastic information and the desire to decrease design lead times, means that there is usually only time allotted for one case to be analyzed extensively. This one datum point to be analyzed has usually been the vehicle after completing the detailed design phase. High-fidelity aerodynamic tools and detailed structural models can then be used in a one-shot approach, aeroelastic behavior correctly assessed, and corrected in an ad hoc manner if required.

With the availability of increased computing power, the number of data points that could realistically be analyzed had greatly increased. This process of assessing aeroelasticity had moved to the preliminary phase. However, to truly be of use and

affect the vehicle design, this aeroelastic information must be available at the conceptual phase. This shift requires large amounts of computing power. Moreover, since the conceptual design phase is a much more dynamic and tightly-coupled multidisciplinary environment, techniques are needed that can manage and render the data useful to the process. For these purposes another field of research is important: MDO (Multidisciplinary Design Optimization) techniques.

As these analysis codes tend to generate considerable amounts of data, designing for aeroelasticity has focused on an *integrated* approach to manage these data streams between codes. This resulted in a number of advanced computer codes, for example NASTRAN (NASA Structural Analysis) [114] and ASTROS [45]. Although this integration approach addresses efficiently the large volumes of data that couple structures and aerodynamics, there are disadvantages:

- Integration does not support very well the alternative of accommodating structures and aerodynamics as specialty groups, autonomous within the design team.

- With more and more disciplines being integrated in these advanced codes, to swap a new higher fidelity code in or change to more detailed models becomes difficult.

- Complex engineering problems cannot be wrapped up in one monolithic code because of the dramatic increase in design variables [103].

MDO research devised ways to decompose complex systems into smaller pieces, usually retaining the disciplinary codes, and allow specialists to regain control over their specific domain. Decomposition of these large complex systems spawned many methods [101]. Common among all decomposition techniques are three key points which differentiate them: the linking of the two levels, the use of approximations, and the definition of objective functions at the two levels.

Because applying decomposition allows these disciplinary codes to be executed autonomously, the way is paved to use statistical tools and create surrogate models. Surrogate models gather enough data to infer an accurate mapping from input to output, subject to validity ranges. Once such a surrogate model is in place, a subsequent change in input does not require a straight execution of the expensive disciplinary code, as long as the new input is within the pre-defined validity bounds [23]. Surrogate models can be in the format of equations in the case of RS (Response Surface), giving a high degree of transparency to the designer: the influence of inputs on the outputs is easily verified. Other surrogate models are based on Bayesian techniques and examples are ANN (Artificial Neural Network) and kriging. Ultimately, surrogate models efficiently create a new design database ahead of time for a specific vehicle and this process is simplified by decomposition.

2.3 Research Hypotheses and Objectives

The questions stated in the opening paragraph of this chapter can now be answered by stating the research questions, and suggesting hypotheses that will answer these.

Under the current circumstances, comprehensive information about the aeroelastic behavior is usually unavailable at the early design stages where it could materially contribute to important design decisions including the configuration options. When this information becomes available later in the process, the vehicle often reveals deficiencies of the aeroelastic behavior that must be corrected by expensive and time-consuming re-designs and local fixes or even retrofits in hardware already manufactured, because it is too late to revise the configuration decisions.

With current methods there is a drive to integrate more disciplines into one monolithic code. This approach of bringing aeroelastic information into the conceptual design space has not worked very well. A radically different approach is needed. The research will identify a new decomposition approach known as BLISS and will

examine how BLISS could be adapted to aeroelastic design.

1. How can a decomposition approach be used at the conceptual design stage using physics-based codes, as an alternative to the integrated approach?

This question may be answered by the following hypotheses:

Hypothesis 1 – Decomposition techniques in conjunction with physics-based analysis codes can be used on a large-scale, time-expensive problem.
Hypothesis 2 – BLISS can decompose and efficiently manage these data flows between physics-based analysis codes.

The chosen decomposition method BLISS relies on a database of surrogate models.

2. Can RS equations, in conjunction with DOE (Design of Experiments), quickly yet accurately describe the current design point neighborhood?

3. Can the problem converge with wide initial design variable ranges around the design point?

The following hypotheses are posed to address these two questions:

Hypothesis 3 – RS in conjunction with DOE can accurately capture the design space trends using a quadratic model.
Hypothesis 4 – The initial wide ranges and resulting inaccuracies do allow for convergence of the optimization to occur.

Having established a need and a new methodology for guidance in airplane design with the consideration of realistic constraints was not enough. The results of this new tool must extract the pertinent data and map them to a design environment if the aeroelastic data is to have an impact.

4. Can the flutter and divergence speed be made functions of wing loading and thrust loading, so these can be plotted in a chart?

This leads to the following hypothesis.

Hypothesis 5 – By scaling the wing and thrust of this one BLISS optimized vehicle point design, an aeroelastic constraint line may be formed.

So far the tools described have assumed that loads are accurately predicted or an aerodynamic database exists that are a truer-to-nature representation. Generating correct, accurate information of loads at the conceptual level remains a non-trivial task. Computationally cheaper linear aerodynamics are typically used at the conceptual design stage. Furthermore, it is usually unclear which loading cases are critical for a particular mission. And even if the critical loading cases could be identified with confidence, generally these cases cannot be simulated with linear aerodynamics codes since these occur in the aerodynamic non-linear region. Alternatives such as use of higher fidelity aerodynamic tools are exceptional at the conceptual level for the dramatic increase in time expense.

The research goal was not to describe an all-inclusive method to handle uncertainty. However, the core of the method should allow the capture of the impact of mission conditions such as altitude, Mach number, and g-maneuver on the vehicle aeroelastic properties. A comprehensive uncertainty analysis can then latter expand on this technique.

5. Can the impact of the mission conditions on the aeroelastic characteristics be visualized such that a point may be chosen which is robust to this uncertainty?

This led to the final hypothesis.

Hypothesis 6 – Executing BLISS for different points in the mission allows to see the main effects of these mission characteristics in the thrust loading versus wing loading chart. In such a way, a point design may be chosen that is robust to these variations.

2.4 Thesis Outline

Chapter 3 introduces the concepts of multidisciplinary design optimization techniques, decomposition and surrogate modeling. This chapter describes the theoretical basis and choice of the BLISS method. Chapter 4 is devoted to the calculation of aeroelasticity. Chapter 5 discusses how BLISS can decompose the aeroelastic problem. In Chapter 6 the proposed methodology and its possibilities are summarized and how this method fits in the design process. Chapter 7 describes results of the method as applied to a proof of concept high speed vehicle. Chapter 8 provides a summary of the methodology's results and recommendations.

Appendix A addresses some of the background of integrating frameworks, and is the foundation for selecting the used integrating framework. Appendix B details the models that were used, for reference purposes. Appendix C is dedicated to validation graphs, whereas Appendices D and E are grouping optimization results.

CHAPTER III

MULTIDISCIPLINARY DESIGN OPTIMIZATION

MDO is a methodology to help in the design of complex engineering systems. These systems are governed by mutually interacting yet distinct disciplines at the subsystem level [101]. A decomposition approach is needed because aeroelasticity is a prime example of such a complex system. Aeroelasticity in design is governed by the interaction of the structures, aerodynamics and performance disciplines.

In this chapter, the selection of BLISS is documented, explaining how the preservation of the (disciplinary) specialty group autonomy is achieved and how advantage can be taken of the increased availability of distributed computing networks.

3.1 Decomposition Taxonomy

Decomposition was briefly described in the previous chapter. An introduction to the "lingo" of these methods is needed. Non-partitioned approaches do not decompose the system at all: a straightforward DOE or saturated (D-optimal) design is used to minimize the number of runs and simply wrap around the whole system. However, complex engineering problems cannot always be wrapped up in one monolithic code because of the many design variables [103]. On top of that, if iteration is required between disciplinary codes, partitioning or decomposition techniques can make the problem less computationally expensive.

The selection of a decomposition method has the overall goal of minimizing execution time and problem complexity. Decomposition techniques strive to make the search for an optimized solution manageable by breaking the problem up in subparts.

Some decomposition techniques give the system group in the project total control over *all* variables. This is a major drawback as this makes the system optimization particularly difficult, and requires the speciality teams to give up control over *their* variables. This loss of control effectively reduces these speciality teams to plain analysts. This was a key feature preventing wide adoption across the industry. In order to counteract this opposition, a technique that reflects the autonomy of the disciplinary groups and gives that group a choice of their methods and tools is needed. In doing so, the decomposition technique will remove the detailed variables from the field of view of the system group.

Decomposition techniques can be classified by: the linking of the two levels, the objective functions at the two levels, and approximation techniques used [104]. Some basic decomposition approaches have crystalized over time which will now be discussed. In the concluding pages of this chapter, a summary table for each technique is given.

In the following exposition, CA (Contributing Analysis) refers to the subspace, and usually coincides with the disciplinary codes, although in general, one subspace can be constituted of multiple CA. An optimization routine is symbolically shown in the following cartoons as a dashed diagonal line. The N^2 method is used to indicate inputs and outputs: the former are arrows leading into the boxes from above and below, the latter are arrows leaving the boxes to the left and right. The design vector $X_D = \{X_1, X_2, \ldots, X_N\}$ includes the design variables X_i for each CA_i.

3.1.1 Multi-Discipline Feasibility

This approach is also known as All-In-One or Fixed Point Iteration and depicted in Figure 13. With this technique, there is no decomposition: one single design vector X_D is sent off to the coupled system. The internal MDA (Multidisciplinary Analysis)

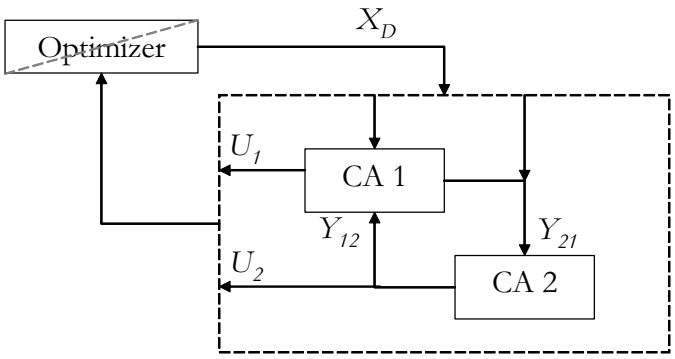

Figure 13: Multi-Discipline Feasibility

is then executed until convergence. The output of CA_i is U_i. This U_i is a multidisciplinary tie with CA_j, in which case U_i can be renamed Y_{ji}. Feasibility is maintained throughout the optimization. However, convergence of this design vector may require many executions of the entire coupled system, usually making the optimization prohibitively expensive. Another drawback of this approach is the requirement for tight numerical tolerance to avoid numerical noise interference. The biggest drawback of MDF (Multi-Discipline Feasibility) is the result dependence on which CA is executed first in the MDA.

The formal statement of the optimization is:

$$\begin{aligned} \text{Find: } & X_D \\ \text{Minimize: } & F(X_D, U(X_D)) \\ \text{Satisfy: } & g(X_D, U(X_D)) \leq 0 \end{aligned} \tag{1}$$

3.1.2 Individual Discipline Feasibility

Optimizer Based Decomposition or IDF (Individual Discipline Feasibility) uses a simple decomposition of the internal MDA and treats the coupling variables Y_{ij} as system optimization variables. The coupling variables, called X_{ij} by the system optimizer,

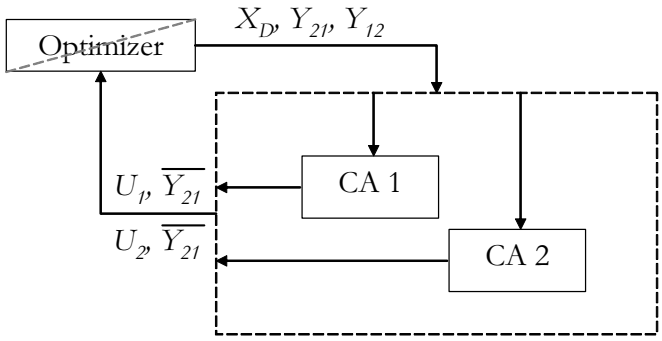

Figure 14: Individual Discipline Feasibility

become undistinguishable from design variables X_D.

As there is complete uncoupling, the CA are now independent of each other and parallel execution of all CA is possible. After execution, the CA outputs U_i and interdisciplinary ties Y_{ij}.

The inclusion of every coupling variable in the system optimization may make this a very large problem, hence IDF is usually only applicable to loosely coupled systems.

The optimization problem is posed as:

$$\begin{aligned} \text{Find: } & X \text{ with } X = (X_D, Y_{ij}) \\ \text{Minimize: } & F(X_D, U(X)) \\ \text{Satisfy: } & g(X_D, U(X)) \leq 0 \\ & J = (U_i - \overline{Y_{ji}}) = 0 \end{aligned} \tag{2}$$

3.1.3 Concurrent Subspace Optimization

CSSO (Concurrent Subspace Optimization) allows subspace optimization, and gives CA more control over local variables. The subsystem level provides results for the optimization carried out at the system level. An important part of optimization is the CA objective function as shown in Equation 3. CSSO simply adds the optimized

CA output to the system objective function in Equation 4. The constraints for each CA are the local constraints g_i, which are directly available within that CA, and a GSE (Global Sensitivity Equation) approximation of the other discipline's constraints g_{approx}. This means other disciplines can contribute to the same constraint in another discipline. The determination of the magnitude of the contribution in the other discipline can be obtained through a SSA (System Sensitivity Analysis) or *responsibility* coefficients.

For each subsystem optimization, two subsets of X exist, namely: X_i which is controlled by CA_i and X_f which includes the fixed variables from the other remaining CAs. The GSE approach guides the optimizers by using local sensitivity information around the current design point. GSE is also sometimes referred to as SSA [100]. The subsystem (CA) optimization is:

$$\begin{aligned} \text{Find: } & X_i \text{ a subset of } X_D = (X_i, X_f) \\ \text{Minimize: } & f(X_i, X_f) \\ \text{Satisfy: } & g_i(X_D) \leq 0 \\ & g_{approx}(X_D) = g + \sum \left(\frac{\partial g}{\partial X_i} \Delta X_i \right) \leq 0 \end{aligned} \tag{3}$$

The system level optimization is:

$$\begin{aligned} \text{Find: } & X_D \\ \text{Minimize: } & F(X_D) \\ \text{Satisfy: } & g(X_D) \leq 0 \end{aligned} \tag{4}$$

The process of obtaining and using the gradients is shown in Figure 15 and Equation 5. LSM, SDV, and LSV are respectively Local Sensitivity Matrix, Sensitivity Derivative Vector, and Local Sensitivity Vector. The LSM and LSV involve partial

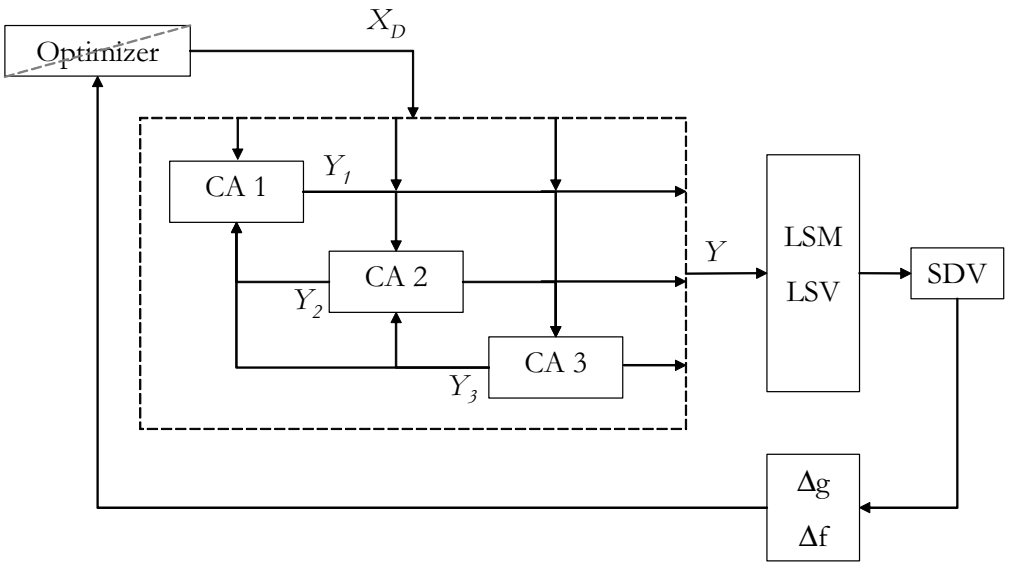

Figure 15: Concurrent Subspace Optimization using Global Sensitivity Equations

derivatives and are easy to obtain. The SDV is computed using inversion. SDV sums up the local effect of the change in the variable (from LSV) and the change of one particular CA with respect to another (from LSM).

$$\begin{bmatrix} 1 & -\frac{\partial Y_i}{\partial Y_j} & \cdots \\ -\frac{\partial Y_j}{\partial Y_j} & 1 & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} \qquad \begin{Bmatrix} \frac{dY_i}{dX} \\ \frac{dY_j}{dX} \\ \cdots \end{Bmatrix} = \qquad \begin{Bmatrix} \frac{\partial Y_i}{\partial X} \\ \frac{\partial Y_j}{\partial X} \\ \cdots \end{Bmatrix} \tag{5}$$

LSM SDV LSV

The use of gradient information in the optimizer has worked well for some research [35]. The computational efficiency of CSSO depends on loose coupling between the codes. If every constraint is influenced by every disciplinary variable, the CA optimization will become an optimization of system level size. The gradient calculation also requires that the CA are lightly coupled. CSSO at the system level encompasses *all* design variables and constraints, and quickly suffers under the dimensionality problem. Grouping all variables in one design vector does solve the coordination

problem since there is none. CSSO is easily used and wrapped around legacy codes. The cost of the approach also depends on the execution time to uncover the sensitivity information and the number of coupling variables this is required for. Matrix inversion requires that the local derivatives be accurate to prevent roll-up errors.

Many problems with CSSO's use of GSE can be solved by an alternative to reflect the coupling in the constraints. The coupling is put in the hands of *responsibility* coefficients [104] which are added variables in the system optimization. These coefficients are used at the CA level to sum the contributions to the total constraint across all CAs simply as the partial derivatives from a sensitivity analysis do. This has an added implication: the system optimization is made more complicated.

3.1.4 Collaborative Optimization

This scheme simplifies the decomposition problem by not including a sensitivity analysis. The design variables are again grouped in one system level vector X_D. The system level proposes values for these variables, and all the CA then try to match these values with a local optimization. So in CO (Collaborative Optimization), the CA is augmented with a local optimization as is depicted by the diagonal line in Figure **??**.

The discrepancy between the local optimization results and system proposed values is captured at the system level by *compatibility* constraints J which are driven to zero. The objective function for the CA minimizes the discrepancy between the system level proposed target values X_i^*, X_f^* for the discipline in question, and the disciplinary calculated output Y_j for this design vector X_i, X_f.

Similar as with CSSO the problem sparsity is key to the operation of CO where the number of local (CA) variables is much larger then the interdisciplinary variables.

The CA optimization is:

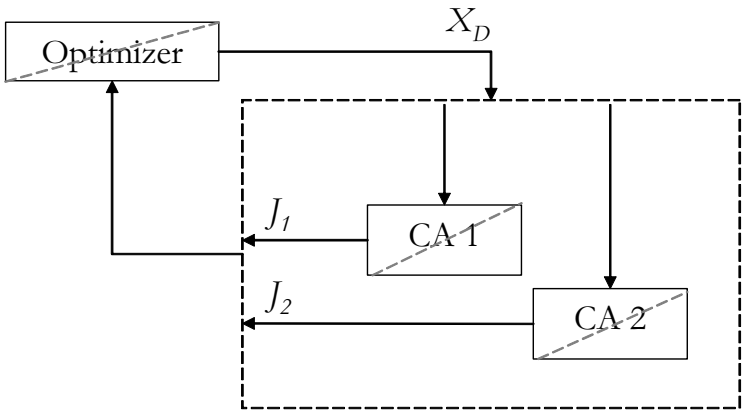

Figure 16: Collaborative Optimization

$$\begin{aligned} \text{Find: } & X_i \\ \text{Minimize: } & J(X_i) = |X_i - X_i^*|^2 + |Y_j - X_f^*|^2 \\ \text{Satisfy: } & g_i(X) \leq 0 \end{aligned} \tag{6}$$

The system level optimization is:

$$\begin{aligned} \text{Find: } & X_D \text{ partitioned as } X_D = (X_i, X_f) \text{ for each subspace } i \\ \text{Minimize: } & F(X_D) \\ \text{Satisfy: } & J(X_D) \leq 0 \end{aligned} \tag{7}$$

3.2 Bi-Level Integrated System Synthesis

The previous discussion of other decomposition techniques has illustrated the challenge of subtask autonomy for closely coupled systems. Decomposing the system and allowing concurrent CA control with each CA influencing all the other ones has proven not to be trivial. BLISS tries to resolve these problems.

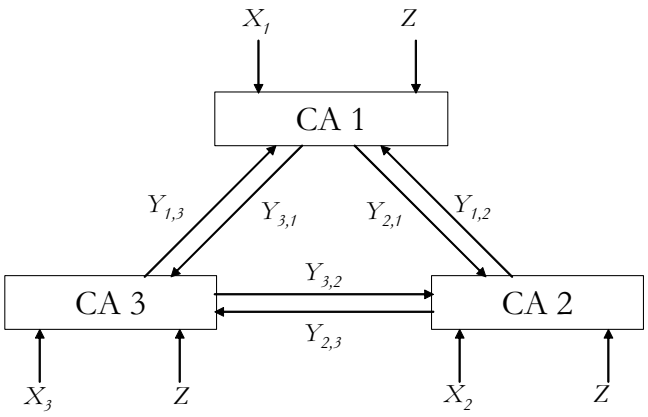

Figure 17: System of Coupled Contributing Analyses

3.2.1 BLISS Decomposition Nomenclature

Using the nomenclature of Sobieszczanski-Sobieski et al. [102] and shown in Figure 17, there is a vector Z which captures the global variables common to at least two disciplines, the X_i vectors are local variables and only of importance to the individual discipline i, and Y_{ij} are the coupling variables with information from discipline j used in discipline i.

The CA outputs will be referred to as hat-matrices ($\hat{}$). The system level generates all CA inputs and are referred to as starred-matrices (*) [18]. By decomposition, each CA works independently, and each CA can be optimized individually and simultaneously to minimize objective function f_i subject to local constraints g_i, all specific to discipline i. The objective function for the i^{th} CA is a linear weighted combination of that CA's outputs used in the other CAs known as Y_k.

The original BLISS version addressed the weight determination in the composite objective function f_i by computing the system sensitivity derivatives of the system objective function F_k with respect to subsystem design variables. This is similar to the use of GSE in CSSO. The partial derivatives are measures of the contribution of each output to the CA objective, f_i.

$$f_i = \sum_{k=1}^{TotalOutputs} \frac{\partial F_k}{\partial Y_k \hat{}} Y_k \hat{} \tag{8}$$

In 2000, BLISS's algorithm was updated with CA objective function weights instead of relying on partial derivatives. The system optimization was given control over these weights, making this w vector a design variable, and was concatenated to the other global design variable vector Z. The CA objective function for the BLISS 2000 algorithm is

$$f_i = \sum_{k=1}^{\text{All Outputs}} w_k Y_k \hat{}. \tag{9}$$

The weights w_k enable system-level control over the emphasis the CA optimization places on Y_k [103]. As Sobieszczanski-Sobieski [102] remarks: Equation 9 mathematically states that the CA should not optimize for any particular, locally selected output, e.g. structural weight, because in a coupled system a particular CA influences the system behavior by means of a number of different output variables.

In general, some of these influences are direct and some are indirect. For instance, if the aircraft range is the objective, the structural weight is a direct input to the range calculation. On the other hand, the wing deformation that is also output from the structures CA only directly affects the aircraft range through the increment of the aerodynamic drag. Hence, locally within the structures domain there is no information whether to optimize the wing structure for increased stiffness resulting in less drag at the expense of a weight penalty or vice versa. BLISS resolves this dilemma by asking the structures discipline to optimize a sum of normalized weight and deformation each weighted by its own coefficient w.

The w coefficients are under control of the system optimization that deals with the system objective, such as the aircraft range, and the system constraints. By controlling the w vector, the system instructs the structural discipline how much emphasis to put on structural weight versus structural deformations (stiffness) for

scoring the best improvement in the system objective. Imperative is the normalizing of Y_k to remove the disparities of the orders of magnitude between the different CA outputs.

As demonstrated by a geometrical proof in Sobieszczanski-Sobieski and Venter [105], the optimization defined in Equation 9 returns a family of designs that are all feasible with respect to the local constraints but different in terms of the output variables contributed to the system. The role of the system optimization is to pick and choose from these diverse designs offered by the individual CA a subset of designs, that together define a system optimized with respect to a particular single objective. The system optimization does that by using the Z variables. In a nonlinear system, a single cycle entailing the round of local CA optimizations followed by a system optimization will not, in general, suffice. In a typical application, BLISS runs through a number of such cycles, and the results are refined from one cycle to the next.

The formal treatment of a local optimization is stated as:

$$\begin{aligned} \text{Given: } & Z, w, Y^* \\ \text{Find: } & X_i \\ \text{Minimize: } & f_i(X_i, Z, w, Y^*) \\ \text{Satisfy: } & g_i(X_i) \\ & X_{lowerlimit} \leq X_i \leq X_{upperlimit} \\ \text{Obtain : } & F_{min}, Y\hat{} \end{aligned} \tag{10}$$

The system level optimizes by changing Z and w, subject to making the compatibility constraints $J = (Y\hat{} - Y^*)$ equal to zero. The compatibility constraints J are the differences between the system optimization results Y^* and the CA output $Y\hat{}$ after evaluation with the current Z and Y^*.

The formal statement of the system optimization is defined by Equation 11 where

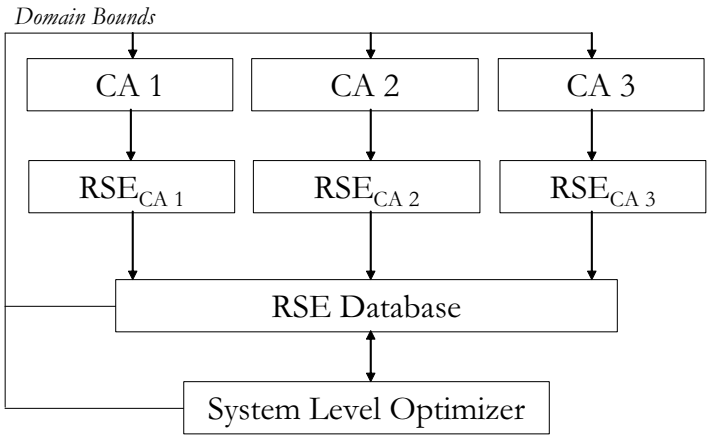

Figure 18: BLISS Response Surface Database

F represents the system design objective function, $Y\hat{}(X_i)$ represents the state variables from the CA, J the compatibility constraints, and g the system design constraints.

$$\begin{aligned} \text{Find: } & Z, w, Y^* \\ \text{Maximize: } & F(Z, w, Y^*) \\ \text{Satisfy: } & g(Z, w, Y^*) \leq 0 \\ & J = (Y\hat{} - Y^*) = 0 \\ & \text{Bounds on } Z \end{aligned} \tag{11}$$

3.2.2 Surrogate Models

In BLISS, the system optimization uses a database of surrogate models to integrate dissimilar CA. Also, for the system optimization to call on the CA whenever new data is needed would be prohibitively expensive and impractical. The surrogate models provide almost instantaneous response at the price of reduced accuracy. After the surrogate models are created, the system level has at its disposition a database of all CA outputs as function of global variables Z, coupling variables Y^*, and weighting

factors w. This concept is shown in Figure 18. Surrogate models may be constructed in a number of ways [56].

3.2.2.1 Response Surface and Designs of Experiments

Response Surfaces rely on an approximation of the response through a set of discrete points dependent on the values of the input variables [17]. When a quadratic polynomial RS is used, Equation 12 must be generated for each $Y\hat{}$ of a particular CA. The CA inputs are $u = \{Z, Y^*, w\}$.

$$Y\hat{} = b_0 + \sum_{i=1}^{k} b_i u_i + \sum_{i=1}^{k} b_{ii} u_{ii}^2 + \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} b_{ij} u_i u_j + \epsilon \tag{12}$$

The b coefficients are determined through a least-squares regression process: b_i are regression coefficients for the first degree terms u_i, b_{ii} are coefficients for the pure quadratic terms u_{ii}, b_{ij} are coefficients for the cross-product terms $u_i u_j$, and b_0 is the intercept. Also, ϵ represents the error introduced by the approximation of a function with a limited set of observations.

The polynomial model approach is not practical when model characteristics are of a high degree, or contain a large number of input variables. For these models impracticality results from the large number of regression parameters needed. In all these cases, other more powerful approaches can be tried. The order of the polynomial required is dependent on the relationship between the input levels and the responses. In general, a quadratic RS suffices and approximates this relationship well for reasonable input variable ranges.

The advantage of the RS equation are the associated field of picking techniques to determine in the most efficient way how the inputs influence the output. These are called DOE tables. The role of the DOE is to propose a logical arrangement of test cases within the design space through which the polynomial RS equation will be fitted. The number of test cases is dependent on the number of variables affecting the

Table 2: Number of Analysis Runs for Several Designs of Experiments

DOE	7 Variables	Equation
3-level, Full Factorial	2,187	3^n
Central Composite Design	143	$2^n + 2n + 1$
Box-Behnken	62	-
D-Optimal Design	36	$\frac{(n+1)(n+2)}{2}$

response, the number of levels that the variable will take, and the desired prediction accuracy of the RS.

A full factorial design makes use of every combination of levels in the design space for all the input variables. Accordingly, such a design requires the maximum number of cases to be run and has the best accuracy. In general, these designs result in a prohibitive number of cases when the design space is large. Therefore, less accurate designs are usually used which combine extreme values of the variables with some center point values. The reduction in number of cases to be analyzed is substantial, as shown in Table 15. The saturated or D-Optimal design represents the limiting case, where the number of points is equal to the number of unknowns, thus not allowing any degrees of freedom for predicting the error term ϵ.

For a standard RS, the equation coefficients $\mathbf{b}$ can be calculated from [84, 78]:

$$\mathbf{b} = \{\mathbf{D}^T\mathbf{D}\}^{-1}\mathbf{D}^T\mathbf{y} \tag{13}$$

where $\mathbf{y}$ is the column matrix of the function values and $\mathbf{D}$ is the design matrix of the DOE. The loss of accuracy intrinsic in RS is to some extent compensated by smoothing of the response that lessens the probability of getting trapped in a local minimum.

For BLISS, the data points the RS equations are fitted to, are generated by repeating the analysis and/or optimization many times in the space of the inputs Z, Y^*, and w.

3.2.2.2 Bayesian Models: Artificial Neural Network and Kriging

The computational unit in a ANN is the perceptron. The perceptron is responsible for mapping inputs onto outputs with a set of weights, biases and a transfer function. These computational units can be linked together into a larger network, a neural network [115].

ANN typically have three or more layers: an input layer, one or more hidden layers and an output layer. The ANN is mainly varied by means of the number of perceptrons in the hidden layer as well as the number of hidden layers. The number of DOF of a network is the sum of the number of weights and biases in the network.

The heart of the ANN is the training process, an optimization procedure responsible for minimizing the errors between the target and simulated outputs by varying the weights and biases. Every point used in the training process is fitted, and a validation process at random points verifies the prediction error of the ANN. The number of runs required to generate one neural network is therefore expensive. An ANN has the advantage of being able to fit multi-modal design spaces. However, there is no clarity as to which inputs are important to the output of the ANN.

Kriging models are chosen to interpolate the data and are fit using maximum likelihood estimation [63]. The kriging method employs space filling experimental designs such as Latin Hypercube variations [55, 61, 107].

The advantageous part of the RS methodology is that the effect of the inputs is very clear and visible through the response surface equation. Kriging retains that by assuming a global polynomial model $f(x)$ (which could be a RS) and a certain departure from that at the sampled points n_s [99]:

$$Y(x) = f(x) + Z(x) \tag{14}$$

This departure $Z(\cdot)$ represents the realization of a localized deviation with assumed mean zero and a certain covariance matrix. This gives kriging models the

possibility to approximate multi-modal spaces, albeit not as well as ANN.

3.2.2.3 Selection Summary for Surrogate Model

Because the present research is focused on the designer, the most transparent method is desired. The RS equation satisfies this requirement: the equation coefficients immediately show the impact each input has on the output. In a quadratic equation this can even include second order effects and cross-product terms with other inputs.

In general, a detriment of these approximation techniques is the limitation to twenty or so design variables [103]. BLISS solved this dimensionality problem by realizing that only a few *global* design variables are important at the system level, whereas there may be many *local* design variables.

Another requirement due to the long execution times of physics-based codes, is the time to construct such a surrogate model. Bayesian models such as Neural Networks require many executions, on the order of thousands. DOE techniques remain the only practical solution in this context: efficient yet accurate for the intended purpose. It is noted that with the continued increase in parallel computing power availability, the cost of DOE and similar methods changes.

3.2.3 Comparison of BLISS with other Techniques

In summary, all decomposition techniques are compared in Table 3. The legend to the table is as follows:

▲ signifies above average qualities of a technique,

▼ signifies below average qualities,

⧫ signifies average qualities, and

∘ signifies unknown or problem dependent qualities.

The important features of BLISS are:

Table 3: Comparing Decomposition Techniques

	MDF	IDF	CSSO-GSE	CSSO	CO	BLISS	BLISS 2000
Computational Effort							
Requires Loose Couplings	▲	▼	▼	▼	▼	♦	▲
Computational Expense	▼	▲	▼	○	○	▼	○
Large System Optimization	○	▼	▼	▼	▼	▲	▲
Conflicting Optimizations	▼	▼	▼	▼	▼	▲	▲
Approximation Techniques							
Numerical Error Possible	▼	♦	▼	♦	♦	▼	♦
Sensitivity Analysis	▲	▲	▼	▲	▲	▼	▲
Linking of Levels	Direct	Dir.	Surrogate	Surr.	Dir.	Surr.	Surr.
Implementation Effort							
Code Modifications	▲	▲	▼	▲	▲	▼	▲
Additional Scripts	▲	▲	▼	♦	♦	▼	♦

- System level:
  - Compatibility constraints are used to link the two levels.
  - BLISS recognizes that there are two levels of variables: system (global) and subsystem (local) variables.
  - The system objective function is the generally to-be-optimized metric.
  - Surrogate models database use RS equations. These have the advantage of smoothing the data – within suitable ranges – so that the design is always feasible when the system optimization needs to be stopped.
- Subsystem level:
  - A composite objective function guarantees that all outputs of a CA that are important to the system level play an implicit role in the system optimization.
  - The system optimization has control over the CA by the weighing parameters in the composite objective function.
  - Surrogate modeling allows for easy integration of the different codes in one design framework.

From the above discussion, the BLISS decomposition method takes advantage of most features in the previous decomposition techniques. However, it is also the most difficult to implement [131]. Some of the difficulties were mitigated by the BLISS 2000 algorithm.

3.3 Conclusions

Engineering system analysis is expensive, time-consuming and a non-trivial managerial task [101]. Two core problems exist:

- At the top-level disparate codes, each with unique data formats and running on different platforms, need to communicate with each other through a system controller whose task is to optimize for a system objective while preserving the couplings among the codes.
- Direct coupling of inputs and outputs between disciplines, which usually result in iterative analysis loops, need to be severed to enable autonomous and concurrent operations at the discipline level.

Two enabling techniques were described to solve this: decomposition and surrogate models.

Decomposition of the tightly coupled aeroelastic problem will rely on BLISS. With BLISS the realization is made that not all variables are important in design. BLISS separates the overall system considerations from the detail level considerations: only accurate data of aeroelastic constraints with respect to global geometry parameters are required.

CHAPTER IV

COMPUTATIONAL AEROELASTICITY

The implementation of BLISS requires knowledge of the coupling variables to decide how these data streams can be approximated. Computational aeroelasticity has traditionally been a notably expensive process. From this perspective, the simulation process has to be streamlined to increase efficiency and accuracy [15]. This chapter will focus on the fundamental calculations and knowledge needed to later intelligently develop the decomposition for maximizing computational efficiency.

4.1 Overview of Aeroelasticity

Aeroelasticity can manifest itself in many ways. All possible combinations are depicted by Collar's triangle in Figure 19. Each aeroelastic effect is a combination of at least two of the following fields: inertia, elasticity and aerodynamics. In this research only flutter and divergence are discussed.

The mathematical treatment of aeroelasticity can be found in many references [11, 13, 16, 31].

4.1.1 The Flutter Mechanism

Flutter can be defined in several ways. Fung described classical flutter as an "oscillatory instability in a potential flow, in which neither separation nor strong shocks are involve" [31]. Most oscillations are stable and dampen out after a short time. Above the flutter speed, the oscillations are manifested by an exponential increase in amplitude, and in most instances, leads to the loss of the vehicle.

The process of wing flutter is physically explained by first considering two different airfoils, each with a single DOF. The first has a rotational, pitching DOF and the

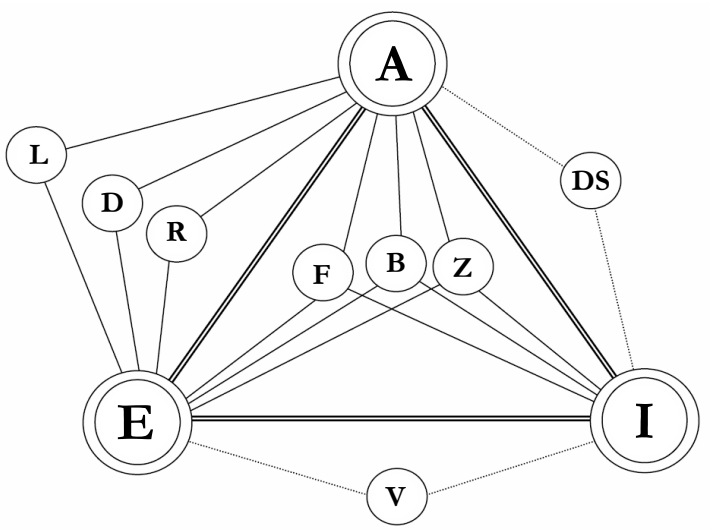

Figure 19: Collar's Triangle

second a translation, heaving DOF.

For the first airfoil capable of rotating around the wing leading edge, consider a torsional spring of stiffness k_θ with no mechanical damping assumed in the connection. If the airfoil is disturbed in still air, the airfoil will oscillate with constant amplitude. This undamped vibration can be described by a sinusoidal function $z(t) = z_0 \cos(\omega t)$, which produces a vortex wake. The number of interacting vortices in the near field behind the airfoil depends on the frequency of oscillation ω. A parameter that is representative of the number of vortices in the wake is the reduced frequency k, a non-dimensional parameter defined as:

$$k = \frac{\omega L}{V} \tag{15}$$

where ω is the harmonic oscillatory frequency, L a reference length, and V the true airspeed.

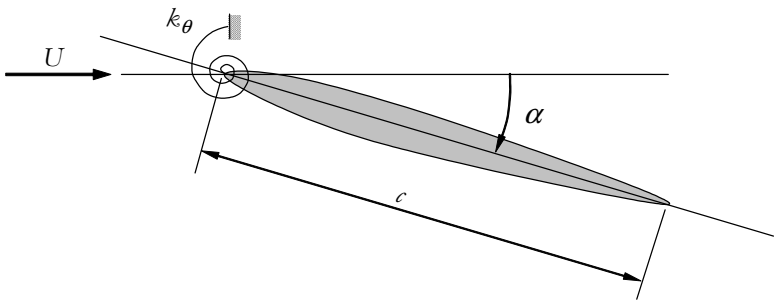

Figure 20: Flutter of Rigid Wing with One Rotational Degree of Freedom

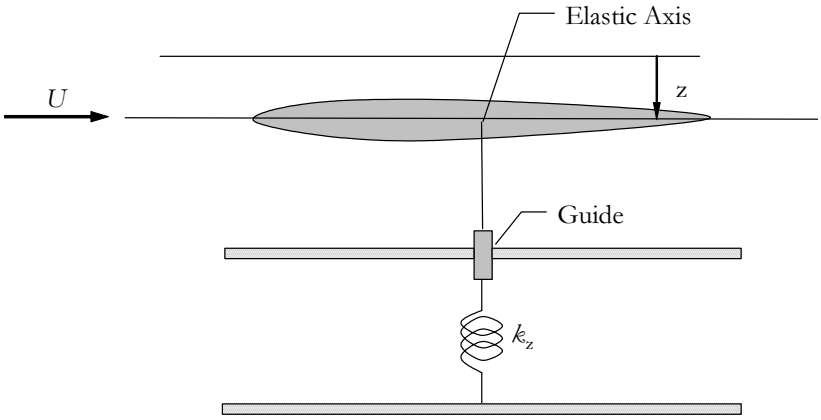

Figure 21: Flutter of Rigid Wing with One Translational Degree of Freedom

Similarly, for the second airfoil with a translational spring k_z, the vertical movement of the airfoil will increase and decrease the angle of attack and this in turn will create a vortex wake in the trail of the airfoil. In both cases the vortices have a damping effect on the oscillation of the airfoil. For this reason, the one DOF airfoils are stable due to the aerodynamic damping, however this is only true in the absence of flow separations or shock waves.

Combining the two DOF onto one airfoil results in the possibility of transferring energy between the two modes. One mode can have a positive or negative effect upon the other one. With one mode reinforcing the other, the translational movement can excite the torsional oscillation due to the unsteady aerodynamic moment. Both modes

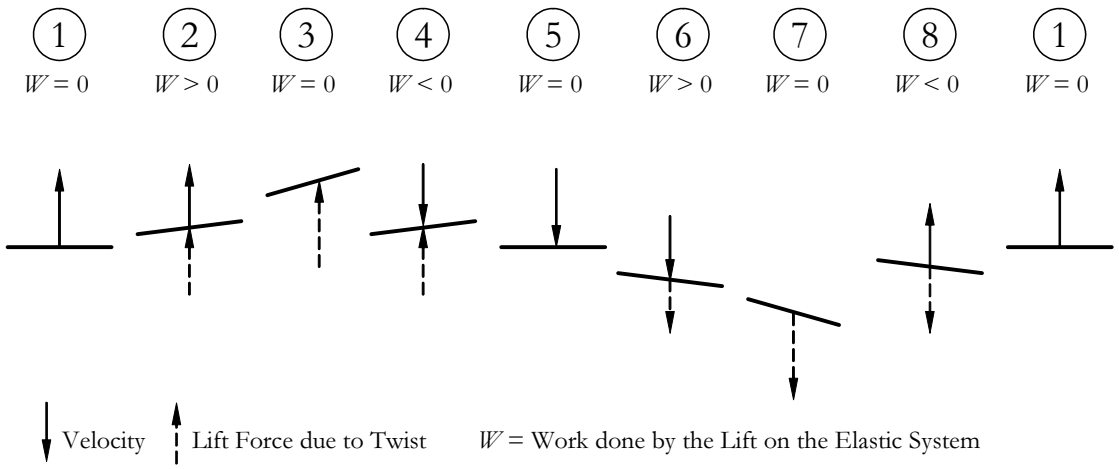

Figure 22: Bending and Torsional Displacements In Phase

are aerodynamically coupled and a disturbance of one of the modes will also excite the other. At low speeds, the oscillations of both modes will be damped without many interactions between the two. At certain speeds, however, the structure of the vortical wake will be such that one of the modes does positive work on the other.

The cartoons in Figures 22 and 23 explain the physical phenomenon. The first shows the bending and torsional displacements in-phase: work done W is positive in one half of the cycle, and negative in the other half-cycle. The net work done is zero, and flutter can therefore not occur. The second cartoon shows a ninety degree phase lag between the flexural and torsional movement. Here throughout the cycle, work done is positive. Although these cartoons are a simplification, these convey the two requirements for flutter to occur: the wing must deform in both bending and torsion, and the two modes must be out of phase with each other. This out of phase motion is defined by a broad range around ninety degrees. Figure 23 therefore illustrates classical flutter.

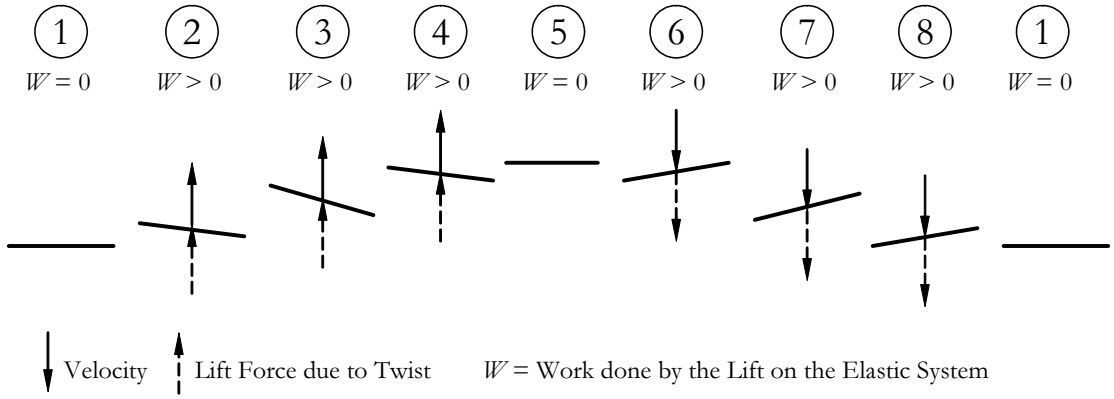

Figure 23: Bending and Torsional Displacements Out of Phase

4.1.2 Determination of Aerodynamic Forces

Before calculating the lift contribution of an airfoil, a reminder is given of the harmonic motion which occurs exactly when the flutter speed V_F is reached. The displacement and speed of the airfoil are described as:

$$z(t) = z_0 \cos(\omega t) \tag{16a}$$

$$\dot{z}(t) = -\omega z_0 \sin(\omega t) \tag{16b}$$

However, a more convenient way to write this harmonic oscillation is by using:

$$z(t) = z_0 e^{i\omega t} \tag{17a}$$

$$\dot{z}(t) = i\omega z_0 e^{i\omega t} \tag{17b}$$

Now, the contribution to the total lift L of an airfoil of width dy under effect of a downward speed v is:

$$dL = {}^1\!/_2 \rho V^2 c a_1 \left(\frac{v}{V}\right) dy \tag{18}$$

where a_1 is the lift curve slope. The speed v is now substituted by $\dot{z}$ from Equation 17b:

$$dL = {}^1\!/_2 \rho V^2 c a_1 \Big(\frac{i\omega z_0 e^{i\omega t}}{V} \Big)\, dy \tag{19}$$

Further introducing and substituting Equation 17a and 15 in the above, results in:

$$dL = {}^1\!/_2 \rho V^2 a_1 (ikz)\, dy \tag{20}$$

An interesting property of performing these substitution is the appearance of the damping component of the translational speed $i\omega t$ in the lift component. This damping is directly dependent on the displacement z and reduced frequency k. There may also be similar damping effects due to twisting, and due to displacement, speed, and acceleration. Hence there are components for $\dot{z}$, $\ddot{z}$, α, $\dot{\alpha}$, and $\ddot{\alpha}$ resulting in:

$$dL = \rho V^2 c \Big\{ (-k^2 L_{\ddot{z}} + ikL_{\dot{z}}) + L_z) \frac{z}{c} + (-k^2 L_{\ddot{\alpha}} + ikL_{\dot{\alpha}} + L_\alpha)\alpha \Big\}\, dy \tag{21}$$

And a similar equation for the moment:

$$dM = \rho V^2 c^2 \Big\{ (-k^2 M_{\ddot{z}} + ikM_{\dot{z}}) + M_z) \frac{z}{c} + (-k^2 M_{\ddot{\alpha}} + ikM_{\dot{\alpha}} + M_\alpha)\alpha \Big\}\, dy \tag{22}$$

Summarizing the aerodynamic derivatives, four groups are noticeable:

1. Stiffness derivatives: L_α gives the magnitude of lift force due to a change in the incidence angle, whilst $-M_\alpha$ gives the moment of this ensuing force. L_z and M_z are small and can sometimes be neglected.

2. Direct damping derivatives: $L_{\dot{z}}$ gives the damping force due to bending, $-M_{\dot{\alpha}}$ gives the damping due to torsion. Both are positive and oppose the motion.

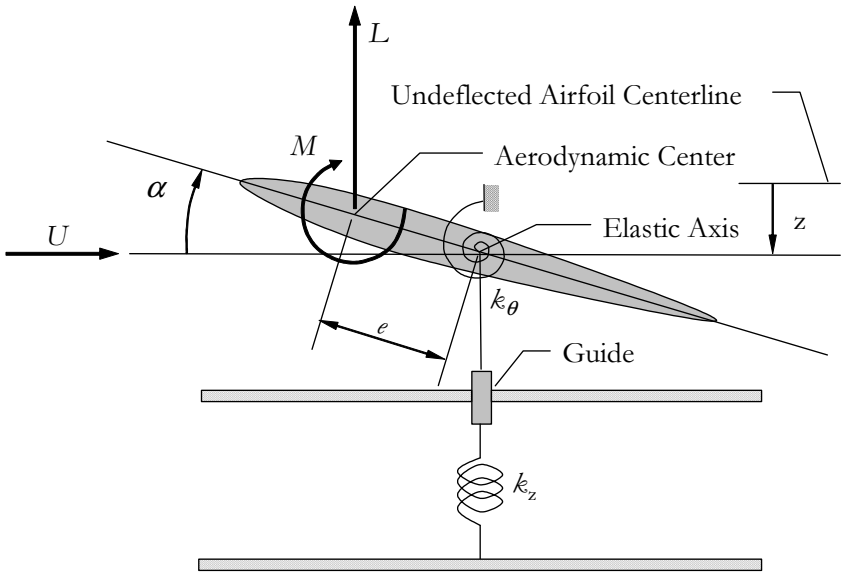

Figure 24: Flutter of Rigid Wing with Two Degrees of Freedom

3. Cross damping derivatives: $L_{\dot{\alpha}}$ gives the lift force due to pitching speed, $-M_{\dot{z}}$ gives the pitching moment due to bending speed. These are usually small, yet desirable to not neglect.

4. Apparent inertia terms: $L_{\ddot{z}}$, $L_{\ddot{\alpha}}$, $-M_{\ddot{z}}$, and $-M_{\ddot{\alpha}}$ represent some addition to the structural inertias.

4.1.3 Derivation of Flutter Speed

For a basic understanding of flutter, the two-DOF three-quarter span (Theodorsen) section of Figure 24 is used.

The following assumptions are used in the analysis:

1. the flutter considered is classical flutter,

2. the structural damping is neglected,

3. the center of mass is located mid-chord,

4. L is the aerodynamic lift, M is the pitching moment about the mid-chord.

The two springs provide a torsional stiffness k_θ with respect to a pure couple, and a stiffness in the vertical translation of k_z with respect to a vertical load F placed at mid chord. If δ is the vertical translation of the spring, then:

$$k_z = \frac{Fc^2}{\delta} \tag{23}$$

where c^2 is used to provide dimensional compatibility with k_θ. If the entire airfoil section is now displaced by a vertical translation z and a wing incidence angle α, then the basic equations of motion become:

$$m\ddot{z} = -L - \frac{k_z z}{c^2} \tag{24a}$$

$$mr^2\ddot{\alpha} = M - k_\theta \alpha \tag{24b}$$

with r the chord-wise radius of gyration about the mid-chord and m the mass of the airfoil section. It is convenient at this time to introduce two non-dimensional parameters q_1 and q_2 that will replace the dimensional DOF z and α:

$$q_1 = z/c \tag{25a}$$

$$q_2 = \alpha \tag{25b}$$

And when substituting these in Equation 24a:

$$mc\ddot{q_1} = -(L + k_z q_1/c) \tag{26a}$$

$$mr^2\ddot{q_2} = M - k_\theta q_2 \tag{26b}$$

Next, the defining equations for dL from Equation 21 is introduced in Equation 26a as substitution for L:

$$\rho V^2 c dy\Big\{(-k^2 L_{\ddot{z}} + ikL_{\dot{z}}) + L_z)q_1 + (-k^2 L_{\ddot{\alpha}} + ikL_{\dot{\alpha}} + L_\alpha)q_2\Big\}$$
$$+ k_z q_1/c + mc\ddot{q_1} = 0 \quad (27)$$

From Equations 17, the following can be defined:

$$\ddot{q_1} = -\omega^2 q_1 = -\frac{k^2 V^2}{c^2} q_1 \quad (28)$$

Equation 27 may now be re-written as:

$$\Big\{ - k^2\Big(L_{\ddot{z}} + \frac{m}{\rho S c^2}\Big) + ikL_{\dot{z}} + L_z + \frac{k_z}{\rho V^2 S c^2}\Big\}q_1$$
$$+ \{-k^2 L_{\ddot{\alpha}} + ikL_{\dot{\alpha}} + L_\alpha\}q_2 = 0 \quad (29)$$

Likewise Equation 26b can be re-written as:

$$\big\{ - k^2(-M_{\ddot{z}}) + ik(-M_{\dot{z}}) + (-M_z)\big\}q_1$$
$$+ \Big\{ - k^2\Big((-M_{\ddot{\alpha}}) + \frac{mr^2}{\rho S c^4}\Big) + ik(-M_{\dot{\alpha}}) + (-M_\alpha) + \frac{k_\theta}{\rho V^2 S c^2}\Big\}q_2 = 0 \quad (30)$$

Equations 29 and 30 are only valid at flutter since a harmonic motion was assumed by using Equation 28. The terms of significance are classifiable in four categories:

1. Inertias: these are the terms prefixed by $(-k^2)$ and may be written as a_{ij} where i and j relate q_i and q_j. The term a consists of a small aerodynamic part and a larger structural part.

2. Aerodynamic damping: these are the terms in (ik) and coefficients are written as b_{ij}. There is no structural damping in this case due to the way the problem was defined.

3. Aerodynamic stiffness: these are the remaining terms which do not have V in the denominator, and are written as c_{ij}.

4. Structural stiffness: these are the terms which have V in the denominator. Since the aim of the problem is to determine the speed at which the flutter condition occurs, it is reasonable to take one of the stiffness terms as a variable $\overline{y}$:

$$\overline{y} = k_z/\rho V^2 S c^2 \tag{31}$$

The remaining term with V in the denominator in Equation 30 then is:

$$k_\theta/\rho V^2 S c^2 = k_\theta/k_z \overline{y} = d_{22}\overline{y} \tag{32}$$

With these substitutions, Equations 29 and 30 become:

$$(-k^2 a_{11} + ikb_{11} + c_{11} + \overline{y})q_1 + (-k^2 a_{12} + ikb_{12} + c_{12})q_2 = 0 \tag{33}$$

$$(-k^2 a_{21} + ikb_{21} + c_{21})q_1 + (-k^2 a_{22} + ikb_{22} + c_{22} + d_{22}\overline{y})q_2 = 0 \tag{34}$$

This system of two equations has four unknowns: q_1, q_2, $\overline{y}$, and k. However, Equations 33 and 34 are complex and thus have a real and imaginary part. Since q_1 and q_2 are complex, these equations can be split in an imaginary and complex one. These two additional equations allow the two parameters q_1 and q_2 to be eliminated from Equations 33 and 34 to give a single equation in terms of $\overline{y}$ and k, yielding an amplitude ratio q_1/q_2. An initial guess on k is required and after an iterative process, a solution to $\overline{y}$ and k is returned. Lastly then, the value of $\overline{y}$ can be substituted in Equation 31:

$$V_F = \sqrt{\frac{k_z}{\rho S c^2 \overline{y}}} \tag{35}$$

which is the flutter speed.

4.1.4 The Divergence Mechanism

The most common type of divergence, wing torsional divergence, is defined as "the speed at which an increment in aerodynamic torsional moment due to an arbitrary increment in twist angle is exactly equal to the increment in elastic restoring torque." This speed is important in the same way flutter was. Below the divergence speed, the structural stiffness is stronger than the aerodynamic twisting moment. This aerodynamic twisting moment becomes bigger with an increase in angle of attack. The speed at which this increase in angle of attack can no longer be counteracted by the structure is called the divergence speed. All aeroelastic flight testing is dangerous, and the vehicle is most likely lost if divergence occurs.

This type of divergence will only occur when the center of twist is behind the aerodynamic center as shown in Figure 25. Design parameters affecting the divergence speed of straight wings are primarily wing torsional stiffness and offset distance between center of twist and aerodynamic center. The increase in torsional stiffness is a costly process at the expense of considerable weight. This is mainly true for isotropic wings; for anisotropic wings, the process of aeroelastic tailoring can be used to raise the divergence speed with a zero weight penalty.

4.1.5 Derivation of Divergence Speed

Only one DOF is needed to consider divergence: the airfoil is restrained in the torsional rotation by a spring k_θ connected at a distance e behind the aerodynamic center. With each change in angle of attack of the airfoil, a certain torque T is created. The torque T acts at the aerodynamic center. This proportional relationship can be expressed as:

$$\theta = C^{\theta\theta}T \tag{36}$$

where $C^{\theta\theta}$ is the flexibility influence coefficient. The torque T is due to the lift and

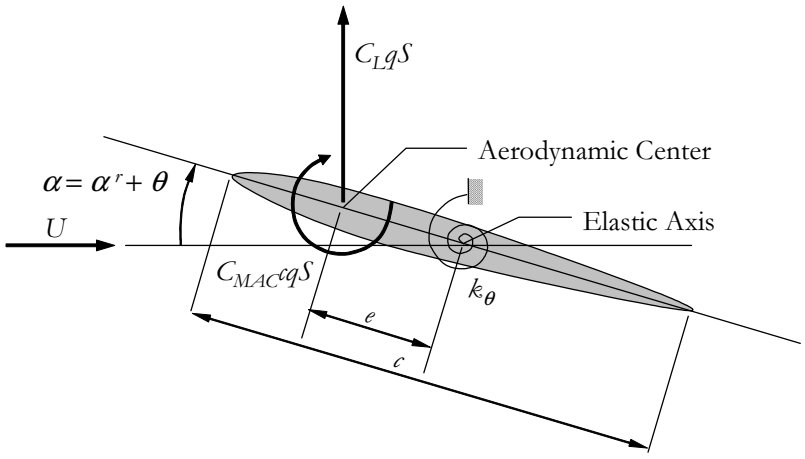

Figure 25: Divergence of Rigid Wing with One Degree of Freedom

moment generated by airflow going over the wing. These two parts can be found back in the equation defining torque:

$$T = C_L eqS + C_{MAC} cqS \tag{37}$$

where C_L and C_{MAC} are the wing lift and pitching coefficient about the aerodynamic center, q is the dynamic pressure, and S the area of a rigid wing sliver. The lift coefficient C_L is measured from a zero lift angle, hence the actual angle of attack α is a combination of the twisting wing angle, θ, and an angle change from the zero lift angle described by α^r:

$$C_L = \frac{\partial C_L}{\partial \alpha}\alpha = \frac{\partial C_L}{\partial \alpha}(\alpha^r + \theta) \tag{38}$$

The elastic twist θ can then be computed by substituting Equation 38 into Equation 37 and combining the result with Equation 36:

$$\theta = \frac{C^{\theta\theta}((\partial C_L/\partial\alpha)e\alpha^r + C_{MAC}c)qS}{1 - C^{\theta\theta}(\partial C_L/\partial\alpha)qSe} \tag{39}$$

No condition has been set preventing the denominator from equaling zero. In case

the denominator does equal zero, the angle θ becomes infinite producing torsional wing divergence. Putting the denominator to zero, the dynamic pressure q_D can be obtained at which this condition occurs:

$$q_D = \frac{1}{C^{\theta\theta}(\partial C_L/\partial\alpha)Se} \tag{40}$$

The resulting divergence speed is then simply:

$$V_D = \sqrt{\frac{1}{C^{\theta\theta}(\partial C_L/\partial\alpha)(\rho/2)Se}} \tag{41}$$

This simple discussion is for an unswept, cantilevered wing. The twist-bend coupling for swept wings has a significant effect on the divergence speed. For backward swept wings this coupling results in a nose-down torsional moment when the wing is bend upward. This has a stabilizing effect on the wing. The bend twist coupling of forward swept wings results in a nose-up torsional moment when bending the wing upward. This has a de-stabilizing effect on the wing and decreases the divergence speed.

4.2 Computational Aeroelasticity

From a computational viewpoint, the general aeroelastic system can be described as:

$$\mathbf{M}\ddot{\mathbf{x}}(t) + \mathbf{G}\dot{\mathbf{x}}(t) + \mathbf{K}\mathbf{x}(t) = \mathbf{F}(t) \tag{42}$$

where $\mathbf{M}$, the mass matrix, $\mathbf{G}$, the damping matrix, and $\mathbf{K}$, the stiffness matrix, are calculated by a FE code, and $\mathbf{x}(t)$ are the structural deformations throughout time. $\mathbf{F}(t)$ can be split into forces arising due to structural deformation $\mathbf{F}_a(\mathbf{x})$ and externally applied forces $\mathbf{F}_e(t)$. The former is calculated using an unsteady aerodynamic computation, and depends on the structural deformations $\mathbf{x}(t)$. $\mathbf{F}_a$ can be considered an aerodynamic feedback as depicted in Figure 26 [20]. The damping matrix $\mathbf{G}$ is

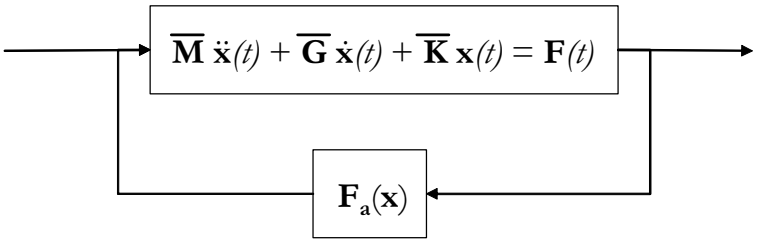

Figure 26: Aeroelastic Closed-Loop System

typically small and may be ignored for simplicity. The externally applied forces $\mathbf{F}_e$ may also be ignored for calculation of stability, so that

$$\mathbf{M}\ddot{\mathbf{x}}(t) + \mathbf{K}\mathbf{x}(t) - \mathbf{F}_a(\mathbf{x}) = 0. \tag{43}$$

Usually this aeroelastic closed-loop dynamic response problem from Equation 43 has to be solved with a CFD time-marching scheme since $\mathbf{F}_a(\mathbf{x})$ is a non-linear function. However, the system can usually be recast as a linear system and solved as an eigenvalue problem. This assumes linearity of the aerodynamic response with respect to structural deformations, which holds for sufficiently small amplitudes. The aerodynamic force feedback $\mathbf{F}_a$ can be written as an aerodynamic transfer function, $\mathbf{H}$, in a convolution integral:

$$\mathbf{F}_a(\mathbf{x}(t)) = \int_0^t q_\infty \mathbf{H}\left[\frac{V}{L}(t-\tau)\right]\mathbf{x}(t)d\tau \tag{44}$$

where q_∞ is the dynamic pressure, V is the undisturbed flow speed, and L is a reference length usually taken to be the chord of the wing. The time marching scheme to solve this convolution integral is computationally expensive.

4.2.1 The *s*-domain

The Equation 44 can be re-cast in the Laplace domain, or *s-domain*. Finding a Laplace transform of an arbitrary function is not easy. This non-trivial conversion of

the aerodynamic transfer function $\mathbf{H}$ to the Laplace counterpart $\overline{\mathbf{H}}$ is a detriment of this domain. In the s-domain, Equations 43 and 44 become:

$$\mathbf{F}_a(\mathbf{x}(s)) = q_\infty \overline{\mathbf{H}} \left(\frac{Ls}{V} \right) \mathbf{x}(s) \tag{45}$$

$$\left[\mathbf{M}s^2 + \mathbf{K} - q_\infty \overline{\mathbf{H}} \left(\frac{Ls}{V} \right) \right] \mathbf{x}(s) = 0 \tag{46}$$

4.2.2 The k-domain

The frequency k-domain relies on the fundamental unsteady aerodynamic parameter, the reduced frequency k, and was useful in simplifying the previous flutter calculations. The aerodynamic transfer function in the k-domain is usually referenced as the *aerodynamic influence coefficient matrix*.

A panel method is used to evaluate the integral at each aerodynamic box which contains a control point with imposed boundary conditions. The assembly of solutions which capture the influence of other panels to control points results in the aerodynamic influence coefficients matrix. This matrix $\mathbf{A}$ relates structural deformation $\mathbf{h}$ to aerodynamic force $\mathbf{F}_h$ so that

$$\mathbf{F}_h = q_\infty \mathbf{A}(ik)\mathbf{h}. \tag{47}$$

The structural deformations $\mathbf{h}$ are different from the structural deformations $\mathbf{x}$ used in previous equations. This is due to the different models used: the aerodynamic model is typically significantly different from the structural FE model. Subsequently a mapping or *spline* is used to translate the structural model deformations to the aerodynamic model. This spline can be represented as a linear transform

$$\mathbf{h} = \mathbf{G}\mathbf{x}. \tag{48}$$

Combining Equation 47 and Equation 48 the aerodynamic feedback $\mathbf{F_a}$ on the structure can be obtained:

$$\mathbf{F}_a = q_\infty \mathbf{G}^T \mathbf{A}(ik)\mathbf{G}\mathbf{x} \tag{49}$$

4.2.3 Modal Modeling

A further simplification is the use of modal modeling. Traditionally, structural optimization requires that for each boundary condition the eigenvalue problem as posed in Equation 46 is solved. This process is expensive because the mass and stiffness matrices are very large due to the thousands of DOF in an actual FE model. Also, non-symmetric decomposition of these large matrices is required [48, 49]. A solution to reduce the size of the eigenvalue problem is the modal modeling approach which is essentially expressed as:

$$\mathbf{x} = \mathbf{\Phi}\xi \tag{50}$$

The modal approach to structural optimization is based on using a set of low-frequency normal modes $\mathbf{\Phi}$ of the baseline structure as a fixed set of generalized coordinates throughout the optimization process [50, 51]. No longer is $\mathbf{x}$ used to describe the deformation of the structure; rather, the modal matrix $\mathbf{\Phi}$ is used. This transformation reduces the order of magnitude of the problem from tens of thousands of DOF to hundreds of DOF. Usually the lowest fifty natural modes are sufficient to describe $\mathbf{x}$. ξ are the generalized coordinates and are the eigenvectors to be determined.

When Equation 50 is introduced in Equation 46:

$$\left[\overline{\mathbf{M}}s^2 + \overline{\mathbf{K}} - q_\infty \mathbf{Q}\Big(\frac{Ls}{V}\Big)\right]\xi = 0 \tag{51}$$

with:

$\overline{\mathbf{M}} = \mathbf{\Phi}^T \mathbf{M} \mathbf{\Phi}$	the generalized mass matrix,
$\overline{\mathbf{K}} = \mathbf{\Phi}^T \mathbf{K} \mathbf{\Phi}$	the generalized stiffness matrix,
$\mathbf{Q}\left(\frac{Ls}{V}\right) = \mathbf{\Phi}^T \overline{\mathbf{H}}\left(\frac{Ls}{V}\right) \mathbf{\Phi}$	the generalized aerodynamic forces matrix.

Equation 51 is referred to as the classical flutter equation.

4.3 Conclusion

An introduction to aeroelasticity described the flutter and divergence phenomena in detail, and illustrated the devastating effect these have on vehicles when trespassed. Due to the mathematical complexities of calculating aeroelastic properties, especially flutter, assessment of it in the conceptual design stage has been impossible.

Discussion of the computational techniques combined with the use of large FE models and detailed aerodynamic models, shows the requirement for simplifying techniques such as modal modeling and the reduced frequency domain. These techniques combined with a decomposition technique from the previous chapter shows promise in further simplifying the aeroelastic problem and bringing the aeroelastic information to the early design stages.

CHAPTER V

BLISS IMPLEMENTATION

Sobieszczanski-Sobieski et al. [103] describe a method for the optimization of complex engineering systems using decomposition. This method shows considerable promise for the aeroelastic problem. The general BLISS depictions from Chapter 3 are recalled and applied to the aeroelastic coupled system in this chapter.

The aeroelastic problem can be cast in the BLISS nomenclature as in Figure 27. The synthesis and sizing code is brought in to referee the conflicts that arise between these disciplines, for example, when structural weight must be traded for aerodynamic drag. The next step is passing the coupling variables Y to the system optimizer, as seen in Figure 28.

After decomposition, the codes can be executed independently and RS equations generated for these codes. The system optimizer can then use this database of RS equations to optimize. The process of creating new RS equations is repeated when the system optimizer tries to violate the validity ranges of these equations as shown in Figure 29.

5.1 Analysis Codes

Computational aeroelasticity in today's environment must be seen both in the light of available computational resources and of a conceptual design endeavor. A FE code was chosen early on for the CSD (Computational Structural Dynamics). From an aerodynamics standpoint, preference goes to either an AIC (Aerodynamic Influence Coefficient) or a CFD methodology. AIC/CSD methods are more suitable in MDO compared to the computationally more expensive CFD/CSD setup. Furthermore,

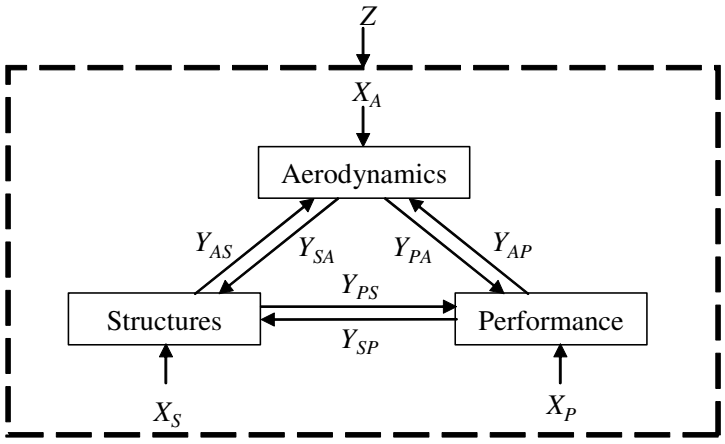

Figure 27: Aeroelastic Coupled System

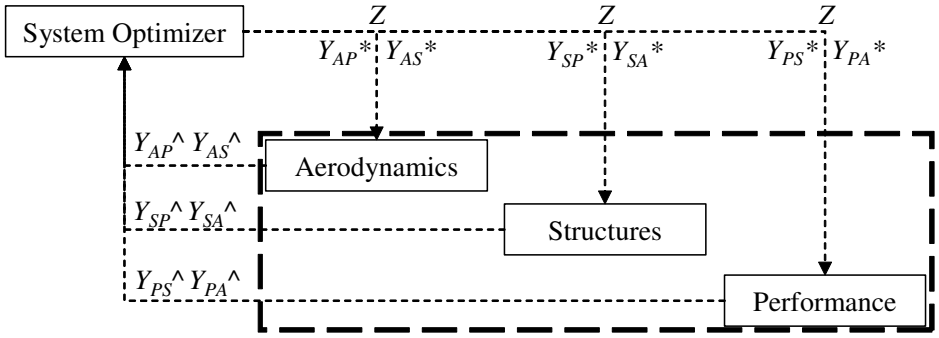

Figure 28: Bi-level Integrated System Synthesis Setup of Aeroelastic System

since this is a *conceptual* design tool, the AIC will provide indication where more accurate solutions with CFD are needed. Chen et al. conclude that "an effective AIC method with sufficient high-fidelity modeling capability remains to be the backbone of computational aeroelasticity" [20]. The analysis codes used in this research are described.

5.1.1 ANSYS

ANSYS is a FE analysis package used widely in industry to simulate the response to structural loading and can be coupled with thermal and electromagnetic effects [8].

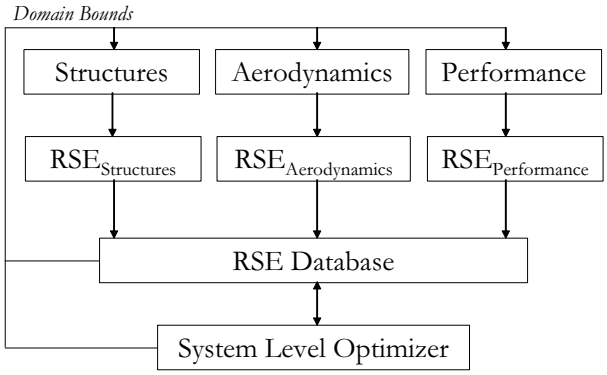

Figure 29: BLISS Flowchart

ANSYS is a general purpose FE program. Structural design optimization is implemented and seeks to determine an optimum design, usually one that is as effective as possible with respect to any objective function the user desires. Any aspect of a design can be optimized by using dimensions (such as thickness), material properties, and shape. Different constraints maybe used such as natural frequency, etc.

5.1.2 ZAERO

ZAERO is a software system that integrates the essential disciplines required for aeroelastic design/analysis [135]. The main features of the ZAERO system are [20]:

- A high-fidelity geometry module to model full aircraft with stores/nacelles.
- Flight regimes that cover all Mach numbers including transonic/hypersonic ranges with Unified Mach AIC matrices as archival data entities for repetitive structural design/analysis.
- Matched/non-matched point flutter solutions using k methods.
- State space aeroservoelastic analysis with continuous gust.
- Trim analysis for static aeroelasticity/flight loads.

- Dynamic loads analysis including transient maneuver loads, ejection loads, and discrete gust loads.

- 3D spline module provides accurate FE/aerodynamic displacements.

The ZAERO system does not provide the structural FE solutions; these are imported through externally computed structural free vibration solutions or the normal modes solutions. The Modal Data Importer module of ZAERO is developed to directly process the output files of some commercial finite element programs such as NASTRAN, ASTROS and I-DEAS. For other FE codes, a "free format" is available that processes the FE output file to obtain structural grid point locations for spline, the coordinate transformations for relating local/global to the basic coordinate system, the modes, the natural frequencies, the generalized mass matrix and the generalized stiffness matrix of the structural FE model.

5.1.3 FLOPS-ALCCA

The Flight Optimization System - Aircraft Life Cycle Cost Analysis code [71] is a multidisciplinary system of computer programs for conceptual and preliminary design and evaluation of aircraft concepts. FLOPS-ALCCA consists of the following primary modules:

- The weights module uses empirical equations to predict the weight of each item. Centers of gravity and moments of inertia can also be calculated for multiple fuel conditions.

- The aerodynamics module uses an empirical drag estimation program to provide drag polars for synthesis and sizing calculations. This module includes smoothing of the drag polars, more accurate Reynolds number calculations, and the inclusion for skin friction calculations.

- The engine cycle analysis module provides the capability to internally generate an engine deck consisting of thrust and fuel flow data at a variety of Mach-altitude conditions.

- The propulsion data scaling and interpolation module uses an engine deck that can be input or has been generated by the engine cycle analysis module.

- The mission performance module uses the calculated weights, aerodynamics, and propulsion system data to calculate performance.

- The takeoff and landing module computes the all-engine takeoff field length, the balanced field length, and the landing field length. The approach speed is also calculated. The module also has the capability to generate a detailed takeoff and climb-out profile for use in calculating noise footprints.

- The cost analysis module in combination with ALCCA uses configuration, engine, performance and weights data from other modules. This module contains the capability to calculate manufacturer/airline costs, production and RDTE (Research, Design, Testing, and Evaluation) costs vs. quantity comparisons, manufacturer cumulative and annual cashflow, and manufacturer/airline return on investment.

5.2 Aeroelastic Analysis Overview

Before addressing the problem, an overview of the aeroelastic analysis in design is given. When designing an airplane and verifying aeroelasticity at certain mission points, the vehicle first needs to be trimmed for that flight condition. This uses the control surfaces to obtain the desired angle of attack that will produce the needed airloads to balance the vehicle for a specific load factor and speed. After this trim analysis, the vehicle is checked for aeroelastic compliance to imposed constraints. It

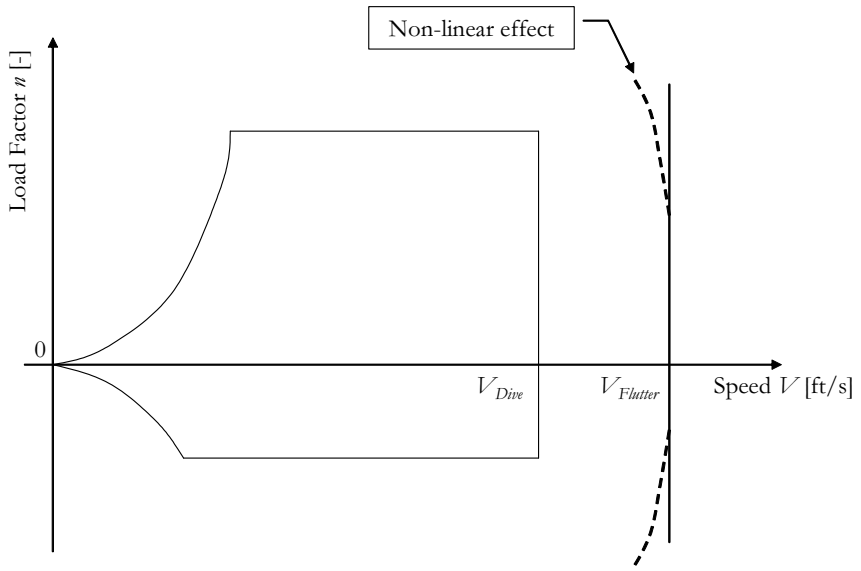

Figure 30: Notional Depiction of Flutter Boundary in V-n Diagram

should be noted that angle of attack by definition has no influence on flutter and divergence speeds. This can be seen by plotting the V-n diagram in Figure 30.

For example, the flutter speed is a constraint line, independent of the requested load factor n. There may be non-linear effects that tend to curve this line. These are due to the increasing airloads on the structure. However, this is an effect not captured by the codes used in the present research.

From Figure 31, the most critical region for an vehicle to experience flutter is in the sea-level, high-subsonic or transonic region. This region is again characterized by non-linear aerodynamic effects. To initially implement the method, a region was chosen were only linear effects play a role. Furthermore, only the main effects of atmospheric conditions will be studied, so a linear main effects approximation across the subsonic, transonic, and supersonic region would lead to erroneous results. The solution was to initially limit the analyzed zone to supersonic speeds as will be shown in Chapter 7.

In the next sections, assumptions will be introduced that will reduce the all-inclusive aeroelastic problem to a tractable size. This slice is what is implemented

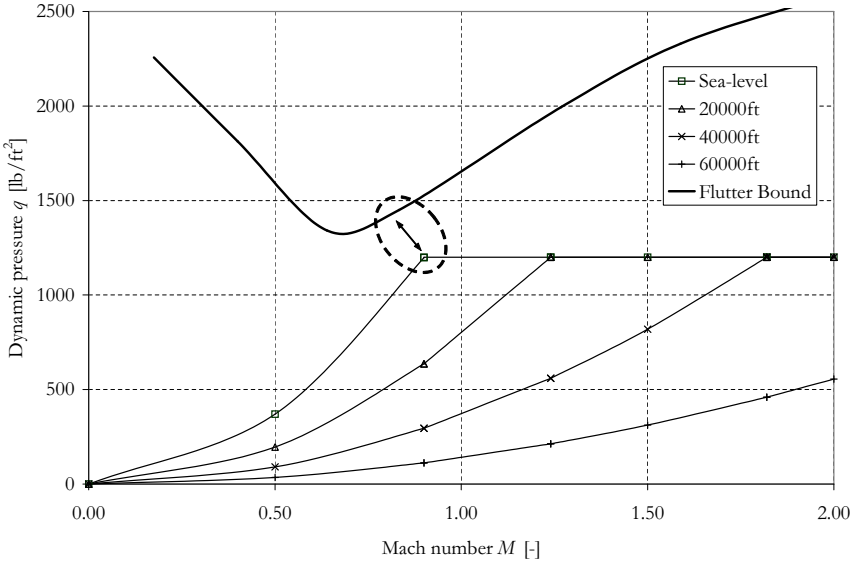

Figure 31: Notional Depiction of Typical Flutter Boundary

in the present research. However, solutions are postulated that allow the full implementation to occur later without having to change the methodology. All these assumptions are summarized at the end of the chapter.

5.3 Aeroelastic Constraint Positioning

Bringing aeroelasticity into the design problem requires calculation of the aeroelastic properties of the vehicle within the loop. Because the vehicle has to satisfy certain criteria, an aeroelastic optimization is required. This aeroelastic constraint can be satisfied at either the system or the subsystem level. These are two distinctly different routes:

1. The vehicle can be aeroelastically optimized while simultaneously performing the structural optimization. This results in all structurally optimized vehicles meeting the imposed flutter and divergence speed constraints. These vehicles are regressed in the RS equations, and this RS is used in the system optimization. The system optimizer will change the geometry to minimize its system

objective, and the process is repeated.

2. The vehicle is structurally optimized and afterwards an aeroelastic analysis of this optimized structure notes the flutter and divergence speeds. RS equations are made for these vehicles. The system optimizer using these equations, then changes the geometry since the flutter and divergence speeds are most likely not meeting the pre-imposed minima. The system optimizer will adjust the geometry trying to meet these minima and this process is repeated until these are met, and the overall system objective has converged.

The two approaches should result in the same optimized vehicles after the system optimization has converged completely. Tracking the aeroelastic constraint at the subsystem or system level has different implications: initial simplifications can be made with the latter approach. These two distinct approaches are illustrated by example of two codes.

5.3.1 Disciplinary-Level Aeroelastic Constraint

In the first case, the aeroelastic/structural optimization code ASTROS would be used. This code has similar capabilities as ANSYS, though using FE methods similar to NASTRAN for the structural analysis. Among its features are analytical sensitivity analyses, constraint deletion concepts to limit the number of sensitivity calculations. ASTROS also has an incorporated panel method to perform aerodynamic analysis and aeroelastic optimization all within the code.

Leaving the synthesis and sizing code aside for now, there are two major data flows: the modal model and the airloads. These are illustrated in Figure 32.

The modal model includes structural dynamic data required by the aeroelastic equations. This link is vital information for these equations as was shown in the previous chapter. These data will need to be approximated by the RS equations, which will be addressed later in this chapter.

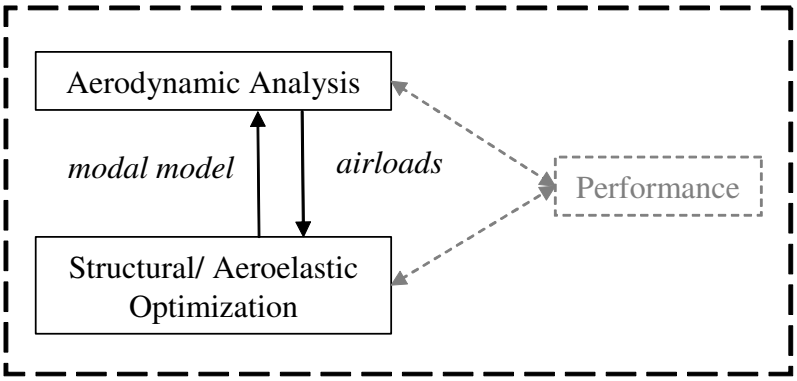

Figure 32: Aeroelastic Constraint with ASTROS

The airloads are calculated by the code ZAERO. These data must be transferred to the aeroelastic/structural optimization code, although the data consist of more then just forces on nodes. As the aeroelastic equations require the $\mathbf{Q}(ik)$ in the space of Mach M and reduced frequency k, there is more than one of these matrices that need to be wrapped up in RS equations.

What is the format of this $\mathbf{Q}(ik)$ matrix? While $\mathbf{A}(ik)$ relates the forces to displacements, $\mathbf{Q}(ik)$ relates the generalized forces to generalized modal displacements. So the i,j element of the $\mathbf{Q}(ik)$ matrix is the generalized force in the i^{th} mode due to modal displacement of the j^{th} mode, or, the j^{th} column in the $\mathbf{Q}(ik)$ matrix is the vector of generalized forces in all modes due to a modal displacement of the j^{th} mode. Therefore, the elements of the diagonal are expected to be large, indicating that motion involving a particular mode will generate a large force in that mode. The other elements do not have to be in decreasing order from the diagonal outside. For example, it is possible that a modal motion of the fourth mode creates a force in the first mode that is larger than the force due to motion of the third mode. It is more the nature of the modes than their number that will dictate the magnitude of the $\mathbf{Q}(ik)$ element.

When this matrix needs to be approximated two variations are deemed possible:

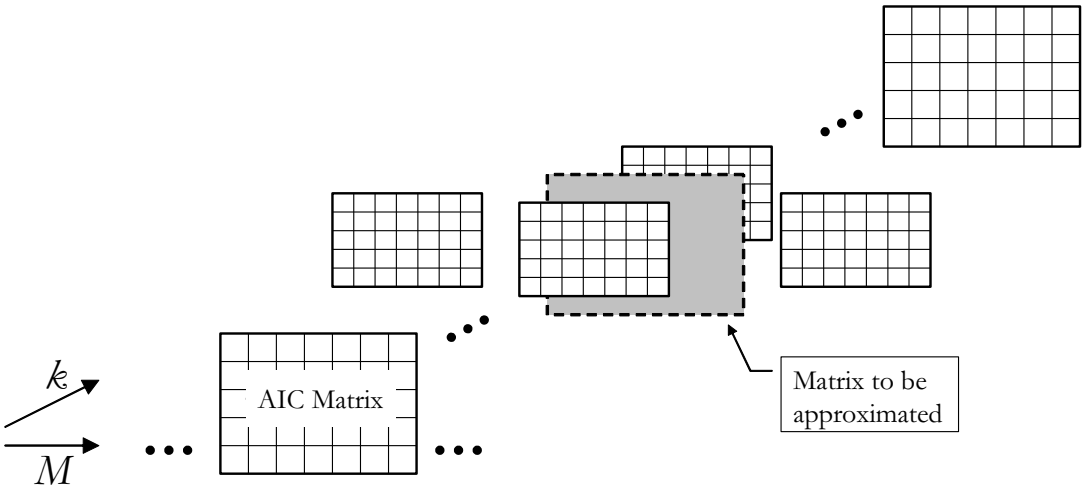

Figure 33: Database Approximation of AIC Matrix

database storage and the RS method. These are utilized as follows:

- Database storage records the corresponding matrix for each set of variable settings. The full matrix with all its elements is saved in a space of M and k. When needed, two one-dimensional approximations are performed, and each element is individually interpolated. This process is illustrated in Figure 33.

- RS methods rely on generating RS equations for each individual element of the matrix. For example, when ten modes are considered in the equations, a ten-by-ten matrix is needed and one hundred RS equations would describe each element as a function of the inputs M and k. This process is illustrated in Figure 34.

Because static aeroelasticity has the reduced frequency k set to zero, only one corresponding $\mathbf{Q}(ik)$ matrix is needed. Flutter calculations require multiple $\mathbf{Q}(ik)$ matrices in the space of Mach number M and reduced frequency k [45]. The elements of the matrix are also complex numbers.

The uncertainty of both approaches lies in the accuracy. The effectiveness of these interpolations will depend on the matrix conditioning, i.e. for a poorly conditioned

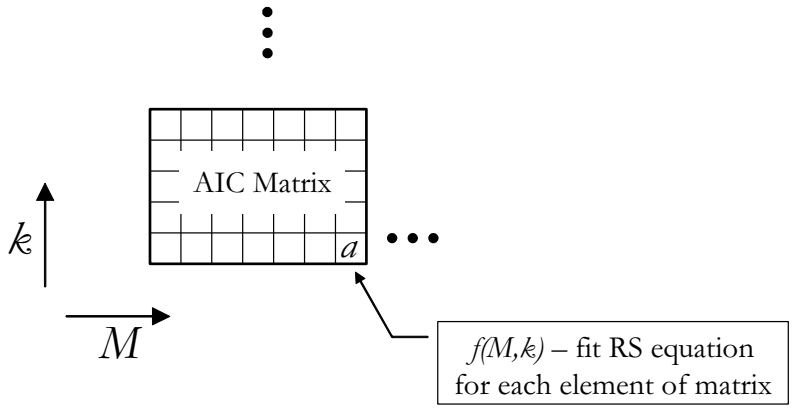

Figure 34: Response Surface Approximation of AIC Matrix

matrix even small errors in its elements might cause a large error in the matrix properties such as the determinant and the eigenvalues that the matrix contributes to the whole analysis process. A measure of ill-conditioning of a matrix is the *condition number*, the ratio of largest singular value over the smallest of that matrix. This number measures the sensitivity of the solution of a system of linear equations to errors in the data and is indicative of the number of decimals susceptible to uncertainty. Values of the condition number near unity indicate a well-conditioned matrix [115]. Experimentation is necessary to illustrate if $\mathbf{Q}(ik)$ is sensitive to being ill-conditioned.

5.3.2 System-Level Aeroelastic Constraint

The alternative approach sends the aeroelastic speeds to the system level. Since the aeroelastic equations and aerodynamic code are grouped into one unit as shown in Figure 35, the structural code only needs to perform a pure structural optimization. The structural code is now only concerned with outputting dynamic data (eigenfrequency and eigenmode) to the aeroelastic/aerodynamic code.

In this case, the airloads from aerodynamics to structures are *only* airloads, i.e. forces on nodes. These can easily be approximated as a pressure distributions since the load on a wing is assumed to vary smoothly from root to tip and from

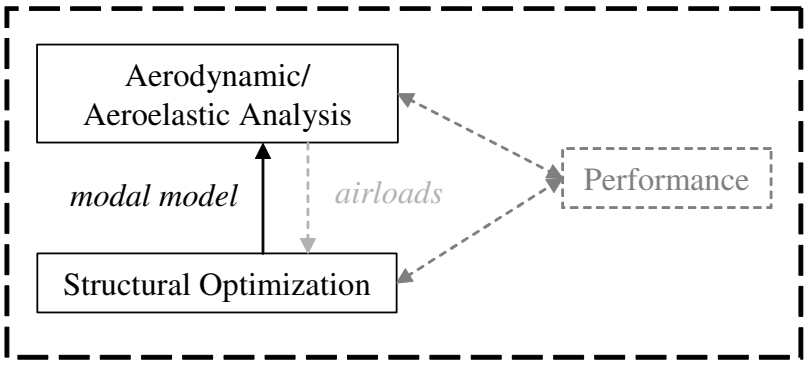

Figure 35: Aeroelastic Constraint with ANSYS

leading to trailing edge as shown in Figure 36 and 37. A few field variables can approximate the pressure distribution, and the constants c are shown in the associated Equations 52 and 53 for a Weibull [43] and elliptic distribution. These five constants c_1 to c_5 can easily be approximated by RS equations.

$$f(x) = c_2 c_1^{c_2} x^{c_2 - 1} e^{-(c_1 x)^{c_2}} \tag{52}$$

$$f(x) = \sqrt{\frac{c_3 - c_4 x^2}{c_5}} \tag{53}$$

Because the structural code outputs the dynamic quantities of the model through a modal model to the aerodynamic/aeroelastic code these properties need to be added to the structural composite objective function. The properties of Equation 54 indicate that the dynamics of the model directly depend on the stiffness matrix $\mathbf{K}$. Therefore, as a first-order approximation of the modal model, the stiffness matrix $\mathbf{K}$ could be used, such that

$$\begin{aligned} \mathbf{\Phi}^T \mathbf{K} \mathbf{\Phi} &= \overline{\mathbf{K}} = \lambda \\ \mathbf{\Phi}^T \mathbf{M} \mathbf{\Phi} &= \overline{\mathbf{M}} = \mathbf{I}. \end{aligned} \tag{54}$$

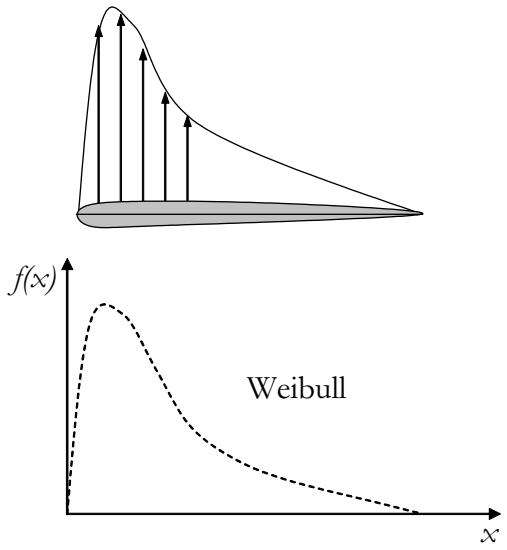

Figure 36: Chordwise Pressure Distribution Field Variable Approximation

A potential problem arises with the stiffness matrix's high dimensionality. Direct inclusion of this matrix is not straightforward. Hence the following is proposed: the inverse of the stiffness matrix is the flexibility matrix **F**. Flexibility captures the full motion displacements of the vehicle. Thus, rather than using the stiffness matrix, the composite objective function would sum the deformations of the vehicle.

5.3.3 Discussion of Choice

As many constraints as possible should be taken care of at the subsystem (disciplinary) level to remain true to the nature of BLISS. This means that the aeroelastic constraint should be satisfied by an optimization either in the structural or aerodynamic code. A possible solution is using the aeroelastic/structural optimization code ASTROS.

There are a few caveats with this choice. A structural optimization is a difficult process with many design variables. Adding aeroelastic constraints to the structural code not only makes this structural optimization more difficult, but the code is also

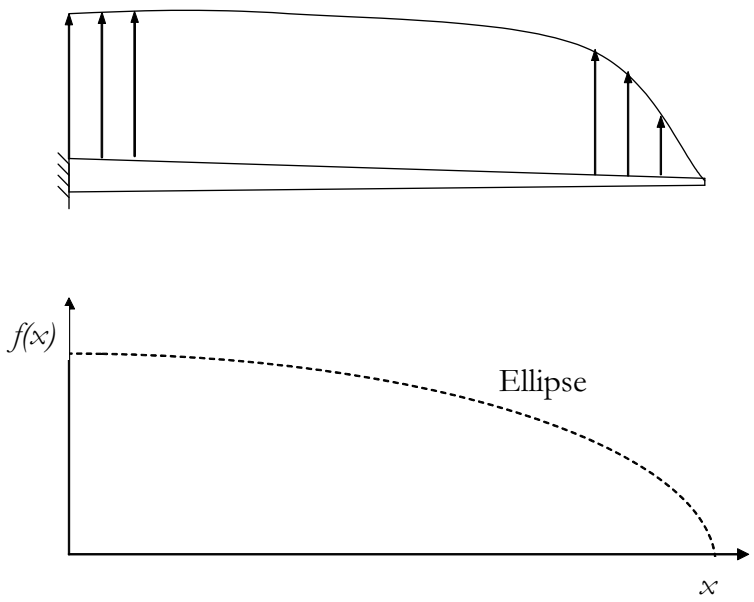

Figure 37: Spanwise Pressure Distribution Field Variable Approximation

more prone to not converge in certain instances. In the alternative, aeroelastic properties of the resulting structure would just be analyzed and aeroelastic speeds simply recorded.

Another trade-off exists between the difficulty of the system optimization versus the execution time of codes. The execution time of the subsystem codes is multiplied by the number of runs in the DOE, and multiplied by the number of RS generations needed throughout the BLISS system optimization. A change in execution time of the subsystem level codes is magnified significantly by this process. Thus, the penalty of additional constraints at the subsystem level is more significant than posed by the alternative. The alternative, satisfying the constraint at the system level, requires perhaps the need for more RS generations, however, the generation is much faster because of shorter execution times. A separate study should determine which is truly faster, however another reason was compelling to satisfy the aeroelastic constraint at the system level.

This research was the first implementation of BLISS in a closely coupled environment with significant data flow among the codes, so the possibility of temporarily leaving out one feedback tie was a major simplification. This allowed proof that the decomposition method works, while noting that the omitted feedback tie could be easily integrated in later iterations of the method. A proposed solution is the use of field variable approximations of the pressure distribution on the structure. An extra pair of RS equations could be made without the need to change or run into limitations of the method.

The airloads coupling to the structural code in Figure 35 can easily be severed while preserving the aeroelastic calculation in the aerodynamics code.

5.4 Couplings to the Performance Module

The performance and economic code FLOPS-ALCCA was brought in as a referee.The program has a sizing algorithm and verifies that the airplane is fuel balanced and can perform the mission. This makes sure that all sized airplanes can fly the mission and are not just conundrums of the aeroelastic/aerodynamic/structural process.

FLOPS-ALCCA takes inputs from and provide outputs for the aerodynamic and structural codes. The more important ones are listed, two possible inputs and outputs exist.

- For most of its analysis, FLOPS uses regressed equations of vehicle properties that are based on weight. A more detailed structural weight breakdown could be supplied from the FE model. Estimating weight of a FE model is not easy though. This is possible but requires innovative algorithms and a database of previously designed elements against which the current model can be calibrated [73]. An example of such a tool is FEMWTS and the process is shown in Figure 38. As these databases are usually proprietary, this link was not included in this design problem.

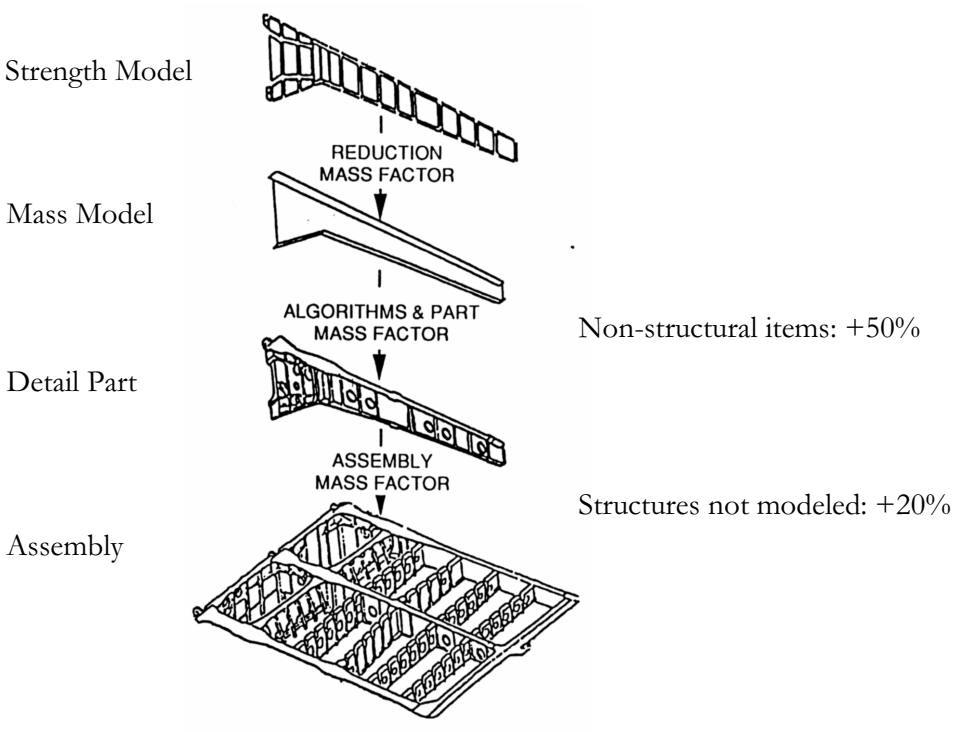

Figure 38: Weight Estimation with a Finite Element Model [73]

- The aerodynamic module in FLOPS consists of drag polars. These are used to fly the vehicle through the mission and determine the amount of thrust and fuel needed based on the generated lift and drag. The FLOPS drag polar for subsonic vehicles are in general based on regressed (historical) data. For most supersonic vehicles this is not possible given the limited amount of data available. Usually the drag polars are generated off-line and included in the model. For the proof of concept, these drag polars existed and there was thus no need to include the process to generate more accurate drag polars. The proof-of-concept baseline was a converged vehicle. Most proof of concept designs for a supersonic business jet have had similar planforms or small deviations of a similar planform. It is possible to include higher fidelity aerodynamics but for simplicity and based on experience of these previous design points, the pre-made drag polars were used.

- The weight of the vehicle that FLOPS calculated after flying the mission could

be fed into a trim analysis. The angle of attack could then be obtained based on the aerodynamics calculated and a trim analysis being performed. The FE model would require that the location and area of the control surfaces be modeled. This addition increases the complexity of the model, adds design variables to the geometry, and increases the number of modes to be included in the flutter analysis. Different flutter modes may now exist such as control surface flutter. This increases execution time of the models analysis and optimization. As a result, the control surfaces were not modeled at this stage of the implementation leaving the angle of attack of the vehicle as a free parameter to be determined by the user.

- As FLOPS performs a fuel balance, detailed knowledge exists of the amount of fuel in the vehicle at any given point in the mission. This implies that a more detailed sizing of structural members is possible if these loads were put on the vehicle.

All four feedback and feedforward couplings are more advanced implementations of the decomposition method to the aeroelastic problem. In addition, these couplings were not included due to the associated increases in execution time and system optimization. The required computational resources for the FE analysis and aeroelastic calculations both drove the implementation to not include these at this stage. A note is made that these couplings could be added afterward and were not part of the proposed method reaching its limitations.

5.5 Implementing BLISS

The decomposition of the system with the aeroelastic constraint is shown in Figure 39. The only remaining feedback variable is the modal model and the decoupled system resulting from BLISS, complete with hat and star variables, is depicted in Figure 40.

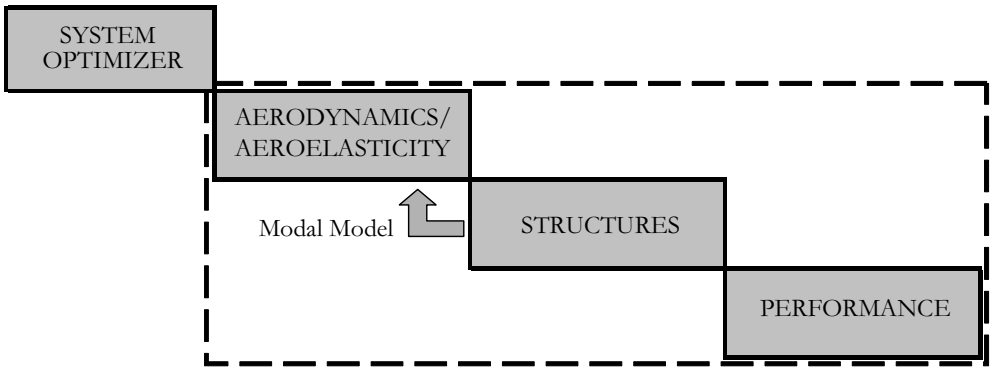

Figure 39: System and Contributing Analysis

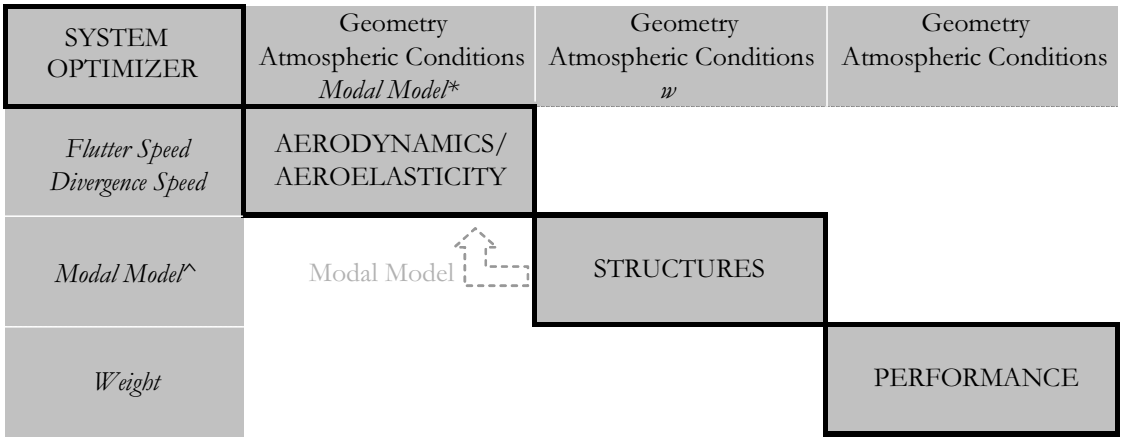

Figure 40: Decoupled System and Contributing Analysis

The global Z design variables are geometry and atmospheric conditions. Combined with this Z vector are the composite objective function weight w and the system optimizer generated Y^* for the aerodynamic/aeroelastic code. This Y^* is renamed $Modal\ Model^*$.

The outputs are listed on the left of Figure 40. These are the aeroelastic speeds for flutter and divergence from the aerodynamic/aeroelastic code. The output for the structural optimization is the modal model which is now a $Y\hat{}$ in BLISS decomposition nomenclature, thus labeled $Modal\ Model\hat{}$. The synthesis and sizing code outputs the take-off gross weight calculated to fly the mission. This weight will be the system

Table 4: Decomposition Detailed Overview

Codes	ModelCenter	ZAERO	ANSYS	FLOPS-ALCCA
Given	$Modal\ Model\hat{}$	Geometry Atm. Cond. $Modal\ Model^*$	Geometry Atm. Cond. w	Geometry Atm. Cond.
Analyze	-	Divergence and Flutter Speed	-	Weight
Find	Geometry w	-	Shell Thickness Beam Cross-section	-
Optimize	Weight	-	Structural Weight + w Deflections	-
Subject to	$Y\hat{} - Y^* = 0$ Divergence and Flutter Speed	-	Stress and Strain	-

Table 5: Response Surface Database

	Aerodynamics	Structures	Performance
Generate RS which output	Flutter Speed Divergence Speed	Modal Model Weight	Range
As a function of inputs	Geometry Atm. Cond. $Modal\ Model$	Geometry Atm. Cond. w	Geometry Atm. Cond. Weight

objective to minimize. All these analyses and optimizations are summarized in Table 4.

Table 5 summarizes the RS equations to be generated. The top row shows what their outputs are as a function of inputs on the bottom row.

5.6 Integrating Framework

The final code left to discuss is the integrated framework software. Many such frameworks exist and for the purposes of this research any one of these would have been sufficient. Phoenix Integration's ModelCenter/Analysis Server was chosen [86]. A discussion of the requirements and different alternative frameworks can be found in Appendix A.

ModelCenter is a tool for automating and integrating codes into a model. Once a model is constructed, optimization and DOE studies may be performed. This is achieved by wrapping the analysis programs, running them in an automated fashion, and linking multiple programs together to form system models.

5.6.1 Analysis Server

Analysis Server enables the incorporation of legacy codes into reusable components which are published on a network. The Analysis Server opens an API (Application Programmer's Interface) allowing control of the complex engineering programs through a simple set of commands, e.g. `get`, `set`, and `execute`.

The Analysis Server is Java-based and runs on many platforms. This allows an analysis program to be wrapped and run on its native platform without modification. Combined with the distributed access capabilities of Analysis Server, stand-alone legacy programs running in different locations on different operating systems may now work together as though modules of a single program.

The process of using ModelCenter in this integrating effort starts with wrapping an analysis program on the Analysis Server. The result is a set of wrapped analysis programs accessible from other networked computers using ModelCenter as a *browser*. Similarly, the Analysis Server is to the engineering world what a web server is to the business world. Instead of sharing html documents, the Analysis Server allows engineering analysis programs to be shared.

5.6.2 ModelCenter

After one or more analysis programs have been wrapped on the Analysis Server, ModelCenter can be used to build a model. An analysis program wrapped on the Analysis Server is referred to as a component. A model is a set of integrated components. Constructing a model involves selecting components in the Server Browser

highlighted as '1' in Figure 41, dragging and dropping these components into ModelCenter. Components are displayed as icons circled with '3' and '4' in Figure 41. Links are displayed as lines between components. Values in the model can be viewed and edited using the component tree, shown as box '2' in Figure 41. The component tree hierarchically displays the model, all of its components, and all the variables in the model [87].

The key features of ModelCenter work very well in combination with decomposition techniques, and especially with BLISS. The most common similarities are:

- ModelCenter automates running analysis programs repetitively. This process is required to run the DOE and eventually generate RS equations.
- Building systems engineering models represented by multiple disciplines into an integrated model can be used to coordinate data flow between disciplines.
- ModelCenter is built around the concept of distributed components. This allows speciality teams to wrap and maintain their programs on their own machines, while other users (system integrators) can access the codes of these speciality teams. This results in more people having access to information that in the past was only accessible to one or two experts.
- ModelCenter provides tools for performing optimization and DOE runs. The DOE tool runs a model through a set of statistically predetermined values to help the user find critical variables in a model. An optimization tool algorithmically tests combinations of values in the model to achieve some objective.

By automating the design process, processes that previously required human intervention in multiple places can now be performed automatically. Building multidisciplinary models is more easily accomplished and the time to execute these is reduced via automation, allowing more designs to be evaluated in a given amount of time.

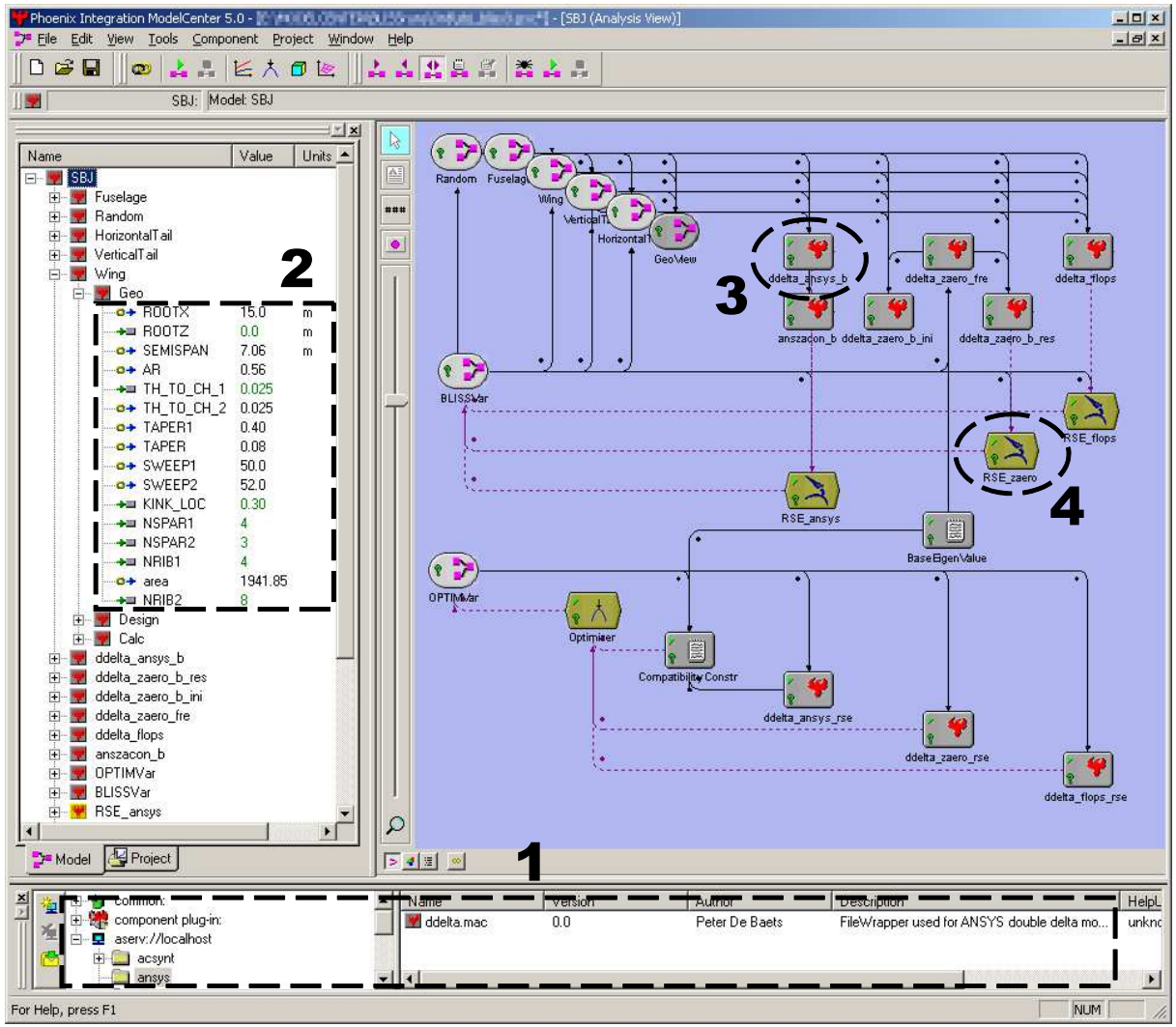

Figure 41: ModelCenter Screenshot Illustrating Components and Model

5.7 Flowchart of Method

The BLISS formulation of the problem consists of two loops: a RS creation phase preceded by an initialization and a BLISS optimization phase using these RS equations. These are described in greater detail below.

5.7.1 Initialization

The RS creation phase started with an initialization step. This initialization updated the structural information to be used by the aerodynamic/aeroelastic code. This update included new eigenfrequencies and eigenmodes with the associated nodal coordinates. These were obtained with a one-shot execution of the ANSYS code using the baseline values.

The template of eigenmodes and structural node locations was required because it was impractical to approximate eigenmodes efficiently. The reason for this was the use of a free mesh to generate the structural model. A free mesh was used because large changes in vehicle concepts were expected. The alternative of using a mapped mesh, generating the same mesh every time but skewed to fit the vehicle mold-line, would eventually result in a bad mesh with large aspect ratio elements. A free mesh allows the structural code to determine the best mesh given a pre-determined maximum size of elements. However, this free mesh resulted in non-fixed, random node locations. If the eigenmodes were to be approximated by RS equations, the eigenmodes were required to be constant. These constant eigenmodes could then be approximated with a few field variables and written as a function of nodal coordinates. Due to the free mesh, writing the eigenmode displacements as a function of continuously changing node locations proved to be non-trivial. This dictated that the eigenmodes were constant during the RS creation phase.

The eigenfrequencies were also extracted but were not fixed. A procedure was

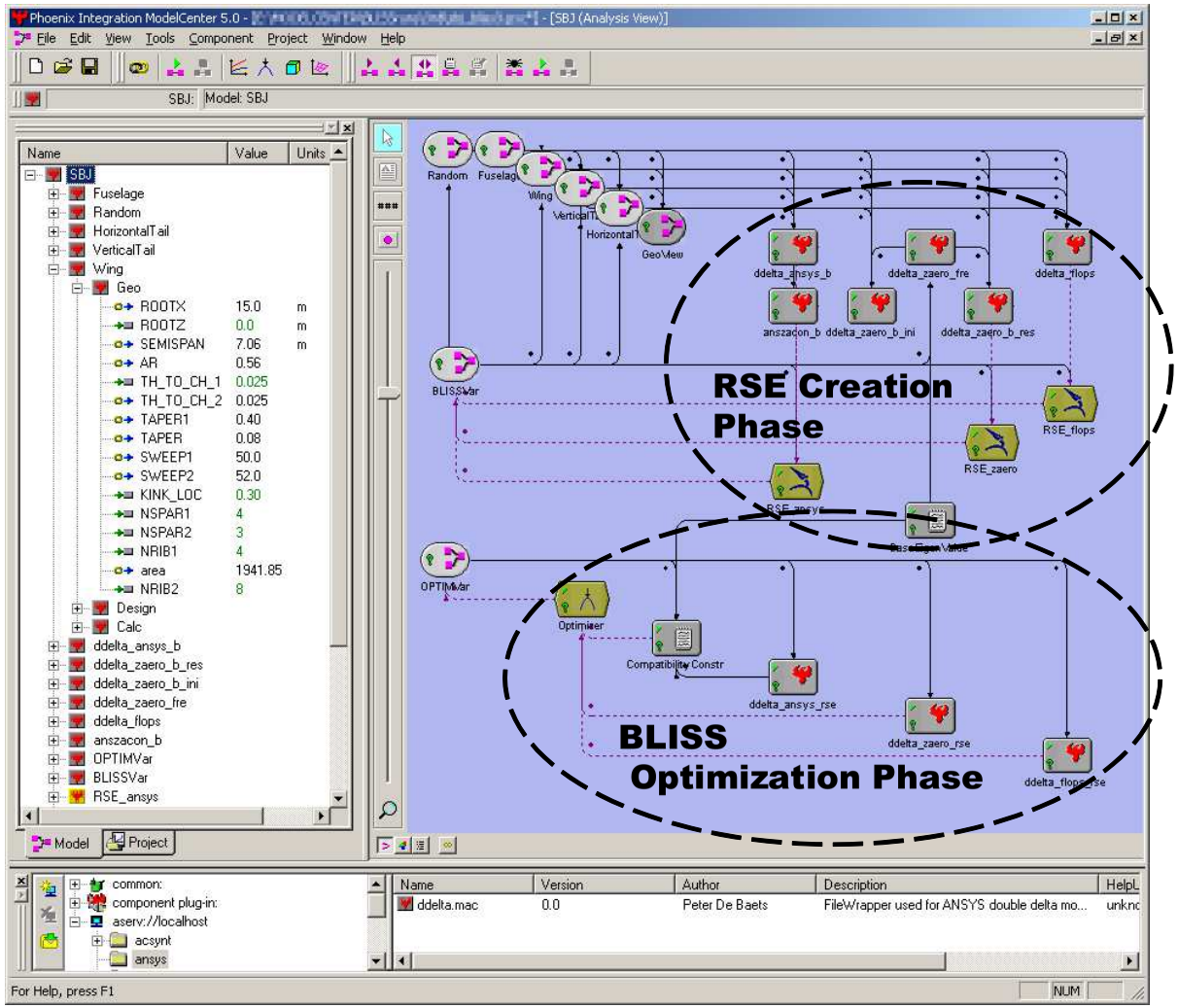

Figure 42: ModelCenter Screenshot Illustrating BLISS Phases

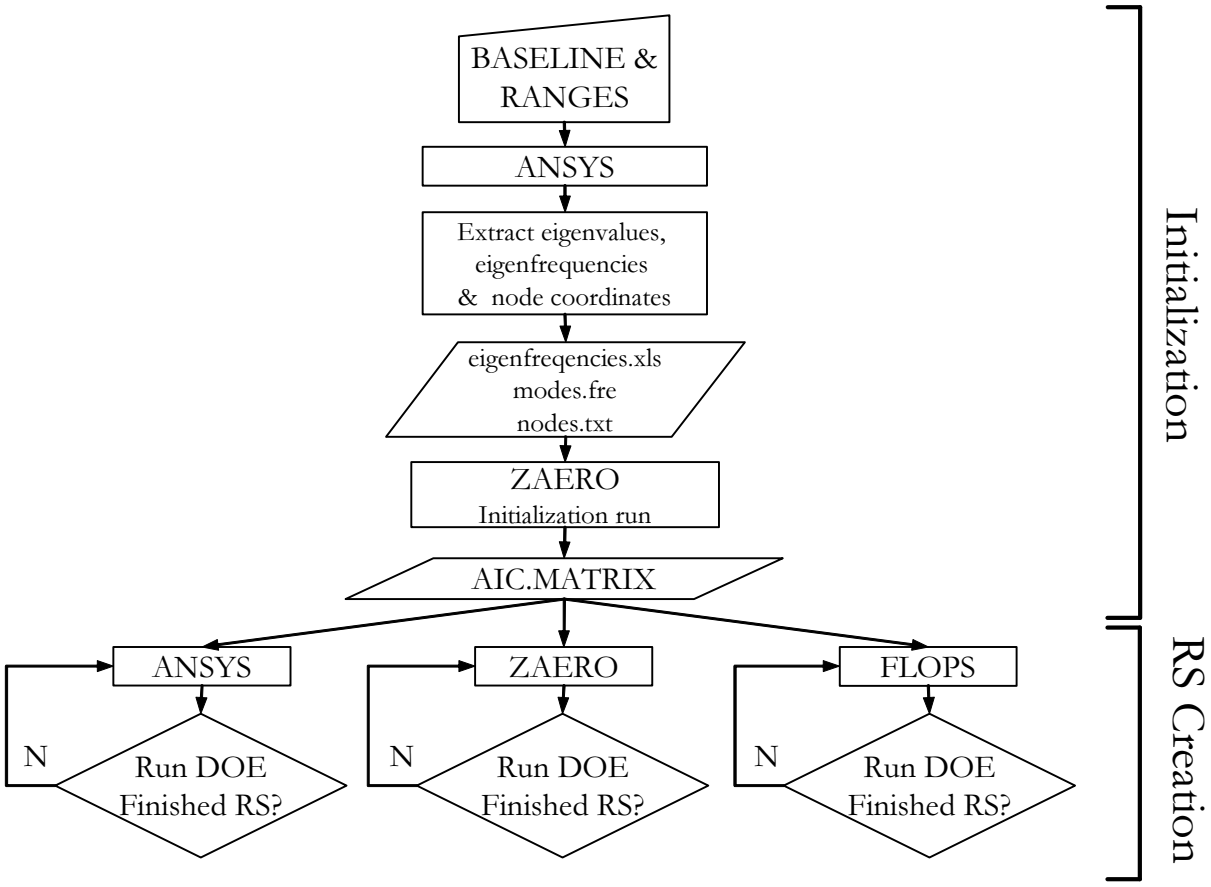

Figure 43: Response Surface Creation Phase

devised to allow for perturbations around these baselines eigenfrequencies. This procedure will be discussed in the ZAERO RS section later.

The extraction of eigenmodes and eigenfrequencies was performed with a modified version of ANSZACON, a code provided by PADT, Inc. [85]. This FORTRAN code was written for ANSYS 5.7, and changes were needed so that it could be used with the new binary format of the latest ANSYS 7.0.

After the extraction of this information, ZAERO was executed to obtain the AIC matrix. This matrix was considered fixed for the RS creation phase since perturbations around a point were performed. This allowed the use of the re-start capability of ZAERO. This feature read in a previously generated AIC matrix, saving execution time. The restart capability is based on the fact that the AIC matrices contain only the aerodynamic characteristics of the configuration. Any changes in the FE model and/or spline input allows for the saved AIC matrices to be reused. However, any changes in the aerodynamic model should force a new AIC matric to be generated. Similar to the baseline eigenmodes used to create the RS, the AIC matrix is also fixed during the RS creation phase to increase computational efficiency.

5.7.2 Creating the Response Surfaces

After the initialization, the RS equations were created. For this purpose a code was used made by SpaceWorks Engineering, Inc., dubbed ProbWorks [106]. This is a wrapped Java-based library of methods. This code was made available on the Analysis Server. After dragging this component in the ModelCenter analysis (shown as '4' in Figure 41), a choice of type of RS could be made. For this research a second order D-Optimal design was chosen and the associated DOE was automatically generated and collected the data from the codes with a simple click of a button.

5.7.2.1 FLOPS-ALCCA Response Surface

The FLOPS-ALCCA RS required no special scripts or changes to the code. FLOPS-ALCCA was wrapped on the Analysis Server, and the execution took on the order of seconds. This was looped through all runs in the DOE table.

5.7.2.2 ANSYS Response Surface

The RS for ANSYS was straightforward: ANSYS' structural optimization was executed for an ANSYS macro structural model for every run in the DOE table. The macro consists of a file that contains all the variables, and the actual file that contains all the ANSYS Parametric Design Language commands to make the structural model. These macro files are printed in Appendix B.

The composite objective function was a summation of structural weight and the summed deformation of the model at various locations. These were the leading and trailing edge of the ribs for the wing, horizontal, and vertical tail; and various points on the fuselage.

After the ANSYS execution, ANSZACON extracted the eigenvalues of the generated ANSYS binary files.

5.7.2.3 ZAERO Response Surface

The ZAERO RS requires some elaboration. Before ZAERO was used, a separate script was executed which multiplied the baseline eigenfrequencies by a factor. This factor was then included in the RS equations, effectively allowing the system optimizer some control over the eigenvalues. The system optimizer used these to minimize the difference between the baseline values and that predicted by the structural RS equations.

The number of eigenmodes that are needed to accurately predict the flutter speed is on the order of fifty [48]. As was discussed in Chapter 3, the number of inputs to an RS is critical in keeping the number of runs in the DOE table low. Only

the inputs determine the amount of runs needed. Including fifty factors in the RS would require a prohibitively large DOE to create these RS equations and therefore become impractical. This property of the DOE is also referred to as the curse of dimensionality.

Two key properties were important:

- All fifty eigenmodes and eigenfrequencies are important in determining the value of the flutter speed. However, only a handful are important in determining the variability of the flutter speed. The effect of a change in the lowest order mode is more important to the variability in flutter speed than the higher-order modes.

- While analyzing different structures, the results showed that most eigenvalues were closely correlated. This meant if a certain eigenvalue changed by a positive or negative amount, then an entire group of eigenvalues would also change by approximately the same amount. This allowed *pooling* of the eigenvalues in a few groups.

In conclusion, the necessary setup of baseline values and pooling the eigenvalues in small groups is a detriment of the free mesh properties and DOE method, respectively.

5.7.3 System Optimization and Testing for Convergence

After initializing the problem and generating the RS database, a system optimization tried to satisfy the minimum flutter and divergence constraint along with the compatibility constraint J while minimizing the take-off gross weight

The compatibility constraints J were the summed absolute differences between the structural RS output for the eigenvalues EV_k and the baseline eigenvalues $EV_{k,baseline}$ multiplied by the pooling factors PF_k for the respective eigenvalue, viz.,

$$J = \sum_{k=1}^{\text{All Eigenvalues}} \frac{|EV_k - EV_{k,baseline} PF_k|}{EV_k} \tag{55}$$

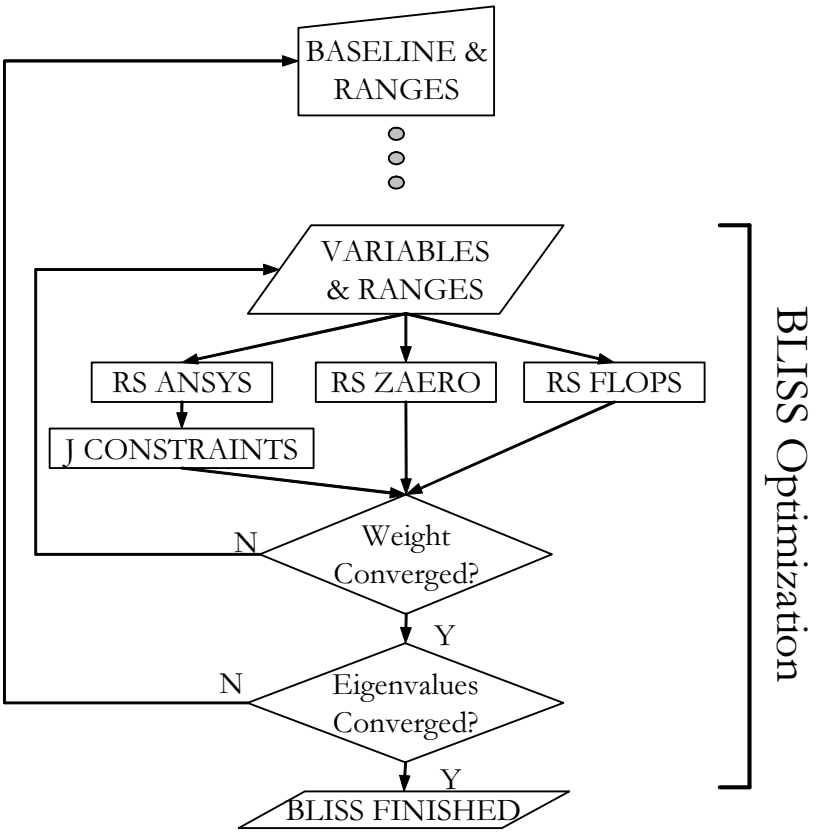

Figure 44: System Optimization Phase Phase

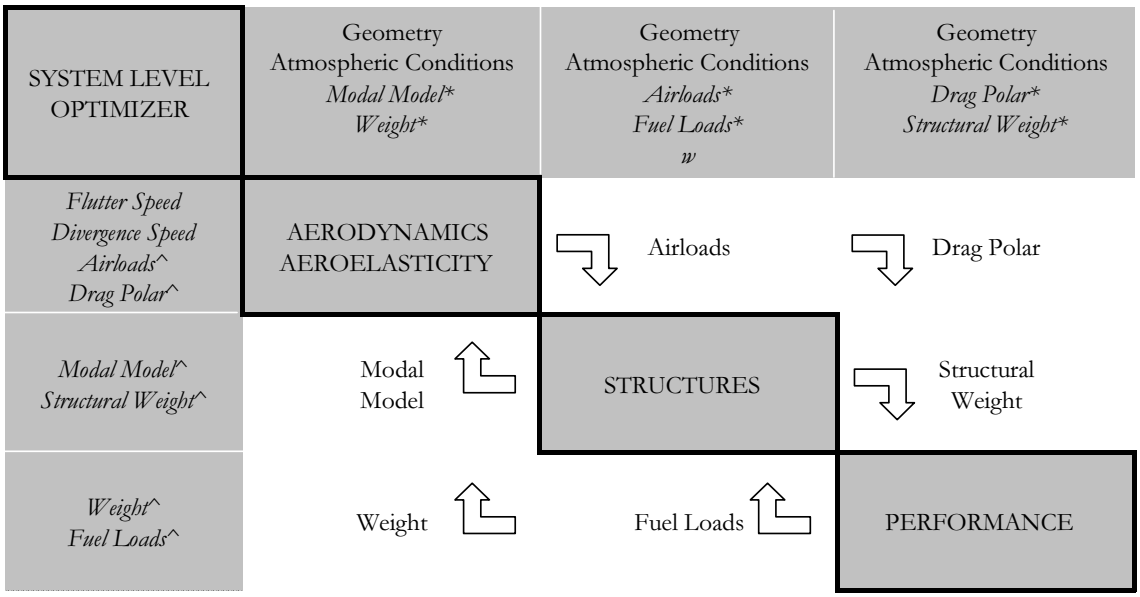

Figure 45: Aeroelastic Problem with All Couplings

By way of the variability and correlation properties, the number of factors to be added could be reduced from fifty to an order of magnitude less. By pooling, there is an error and the resulting solution after each system optimization did not necessarily reflect the true eigenvalues of the structure. Thus, after every system optimization the structural dynamic properties were updated with new baseline values. In this decomposition technique implementation, the coupling manifested itself in this way. As such, Equation 55 mathematically states the coupling that is present between the aerodynamic and structural disciplines.

The system optimizer stopped if the take-off gross weight had converged, or if one or more constraints could not be satisfied after a fixed amount of iterations. The global BLISS convergence test was to check if the eigenvalues had converged (if J had converged to a small value.) If not, a new initialization phase was started and the process repeated.

5.8 Summary of Assumptions

As a summary, the entire aeroelastic problem is shown in Figure 45. The following assumptions were made throughout the chapter to help scale down the initial problem.

- When positioning the aeroelastic constraint at the subsystem level, the airloads vector includes the AIC matrix. The size of this matrix makes it non-trivial to approximate. The proposed solution was to put the aeroelastic constraint at the system level. Then it was proposed to use field variable approximations of the pressure distribution. However, this also allowed that the airloads coupling to the structural code could be severed while preserving the aeroelastic calculation in the aerodynamics code.

- The inline calculation of drag polars from the aerodynamic module was also not included at this point. The proof-of-concept baseline was a converged vehicle as will be shown in Chapter 7. Because all previous research into this vehicle had converged to similar planforms or small deviations of this similar planform, it was decided to fix the drag polar. This was rooted in experience gained from previous designs. Later in Chapter 7, the implication of this assumption is shown.

- A detailed structural weight breakdown could be supplied from the FE model. However, it was argued that estimating weight of such a model was not easy and required innovative algorithms. A database of previously designed elements is needed to do this accurately (e.g. FEMWTS). Since these algorithms did not exist in-house or are proprietary, the coupling could not be implemented.

- The calculated weight from the synthesis and sizing code could be used in a trim analysis in the aerodynamic code. The angle of attack could then be obtained based on the aerodynamics calculated and required trim condition. However,

the location and area of control surfaces is required. In order to model these surfaces, an increase in model complexity is needed. This adds geometric design variables, increases the number of modes needed in the aeroelastic analysis, increases execution time of the analyses, and different flutter modes may now exist. While aileron reversal is probably a design condition for the proof-of-concept vehicle, this condition was not included due to the aforementioned added complexity.

- As FLOPS performs a fuel balance and registers the amount of fuel in the vehicle throughout the mission, these additional loads could be put on the structural model. The result would be a more accurate sizing of the structural members. However, the inclusion of airloads should be the first additional coupling before the more advanced addition of fuel loads.

- Lastly, the modal model coupling was not omitted. This coupling included the structural coordinates, eigenmodes, and eigenvalues. Due to a free mesh used for the structural model, the eigenmodes were kept constant for each RS generation. Pooled eigenfrequencies were allowed to change within certain ranges. After the system optimizer was finished with one iteration and new RS equations were needed, the eigenmodes and eigenfrequencies were updated.

5.9 Conclusions

The aeroelastic problem was decomposed in this chapter. Some assumptions were needed to do this and to simplify the problem. Care was taken that these assumptions could be altered later, and solutions to the assumptions were therefore included.

A detailed flowchart of the BLISS method was presented that shows the data flows and approximations used.

CHAPTER VI

AEROELASTIC CONSTRAINTS IN DESIGN

This chapter will discuss how the BLISS implementation can be used to project the aeroelastic constraints in the designer's space. This will indicate how the results were processed and the followed results strategy after BLISS converged to a solution.

One specific group of variables has not been discussed so far. This group is the atmospheric condition variables added to the global Z design vector. The purpose of these will now be shown.

6.1 Determining the Constraint Line

BLISS by definition generates a point design, i.e. one vehicle is optimized to conform to all constraints. In order to be used in a design environment, a constraint *line* is needed. A notional thrust versus wing loading example is shown in Figure 46 with the one BLISS optimized vehicle. This point has a flutter and divergence speed associated with it as well as other vehicle metrics such as price, emissions, take-off and landing field length, lift over drag, a take-off gross weight, etc.

The procedure to obtain a constraint line involves scaling this point in wing area S and thrust T. The former is done by increasing the span of the wing while keeping constant all other wing parameters such as taper, sweep, aspect ratio, etc. The latter is achieved in the same way by increasing or decreasing the engine, while keeping all other parameters the same. This principle is shown in Figure 47.

At these eight additional points, the scaled BLISS vehicle is simply analyzed, i.e. the structural optimization is executed, the eigenfrequencies and modes are extracted, the aeroelastic/aerodynamic code is used to obtain the flutter and divergence

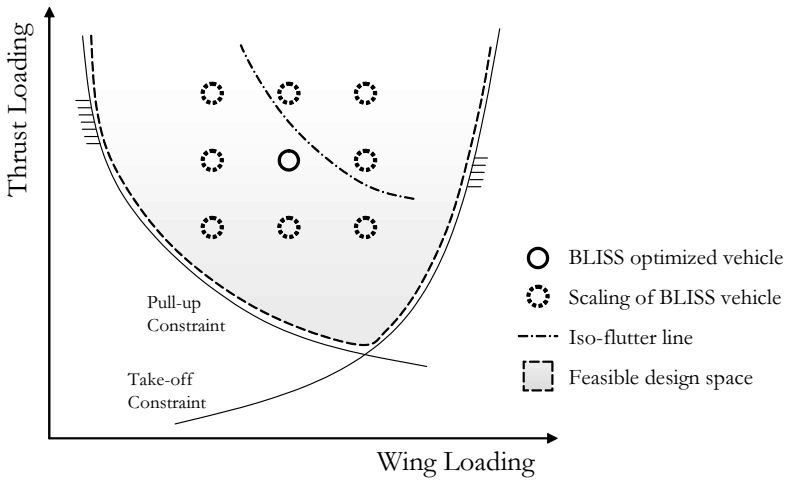

Figure 46: Thrust versus Wing Loading - One BLISS Point

speed, and the synthesis and sizing code to get the performance metrics.

For this research, the flutter and divergence speed are of importance. For example, at these eight points the flutter speed is written down as shown in Figure 48. In this graph, the speeds are normalized with respect to the BLISS optimized vehicle which as a result has a flutter speed of 1.00. The iso-flutter lines for 0.75, 1.00 and 1.50 are drawn. In Figure 46, a fictitious iso-flutter line is also shown. The designer now knows where the flutter, or alternatively divergence, speed minima can or cannot be met.

6.2 Determining the Influence of Uncertainty

Understanding the effect of flight condition on these lines requires that some other free parameters were used. For this research, three parameters were chosen: Mach number M, angle of attack α, and altitude z. In order to understand the main effects of these three variables, a four run DOE is needed. The normalized settings for these parameters are shown in Table 6: '1' means a high setting and '-1' a low setting for the specific variable.

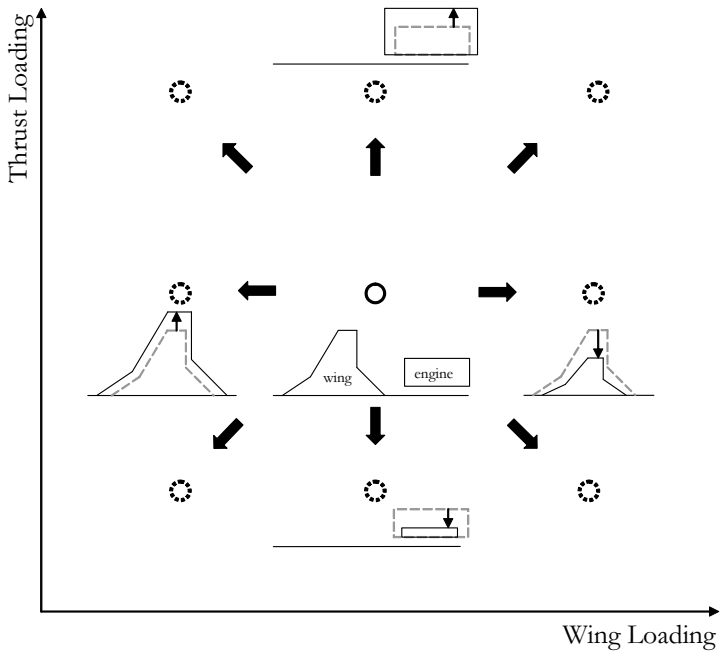

Figure 47: Scaling the BLISS Optimized Vehicle

Table 6: Design of Experiments for Atmospheric Parameters

	M [-]	α [deg]	z [ft]
RUN 1:	1	1	-1
RUN 2:	1	-1	1
RUN 3:	-1	1	1
RUN 4:	-1	-1	-1

After obtaining the result of all four runs, which give four different vehicles optimized for BLISS, the scaling approach mentioned in the preceding section is performed for each case. Four similar plots to Figure 46 can be obtained for each vehicle.

By interpolating between these four plots, the main effect of Mach number or any of the other two parameters can be observed. The result of this is the same constraint, for example a minimum flutter speed, moving with respect to the chosen parameter Mach number, altitude or angle of attack, as in Figure 49. This plot allows the designer to see what trends exist if the airplane were allowed to fly at different speeds, different altitudes, pulling more or less g-forces.

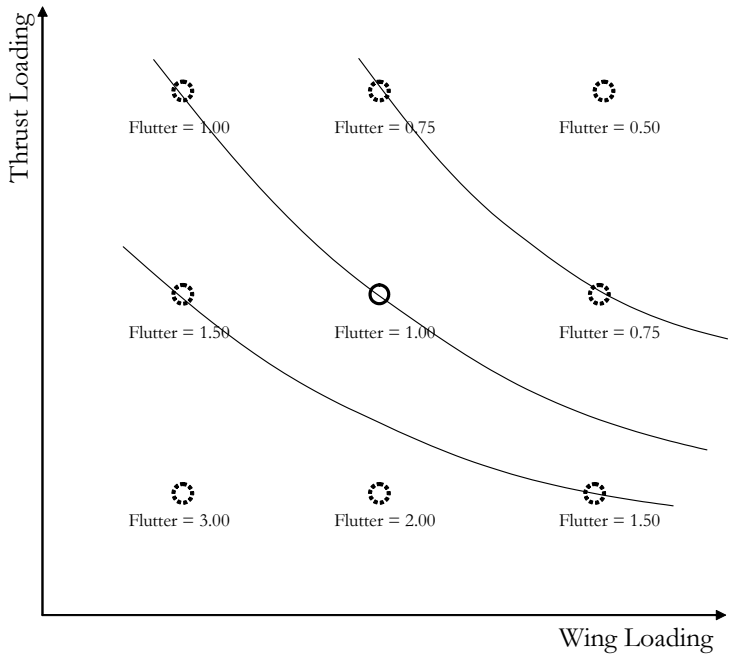

Figure 48: Plotting Aeroelastic Properties in the Thrust versus Wing Loading Graph

6.3 Aeroelastic Decomposition Process Flow

The overall method to project aeroelastic constraints is summarized and illustrated in Figure 50 and 51. The inputs to the method are the atmospheric/mission conditions and geometry baseline values. Within ModelCenter, the BLISS decomposition is executed until convergence. This flowchart is written for an aeroelastic analysis. However additional disciplines may be included in the process as long as these can be decomposed so to run autonomously.

After convergence of the BLISS decomposition loop, one BLISS optimized vehicle is the result. This is one design point, which was minimized for a particular system objective and subject to certain constraints. It follows that the point design is only as good as the number of constraints that were included in the optimization and the fidelity of the codes that provided analysis data.

This one vehicle has attributes and metrics.

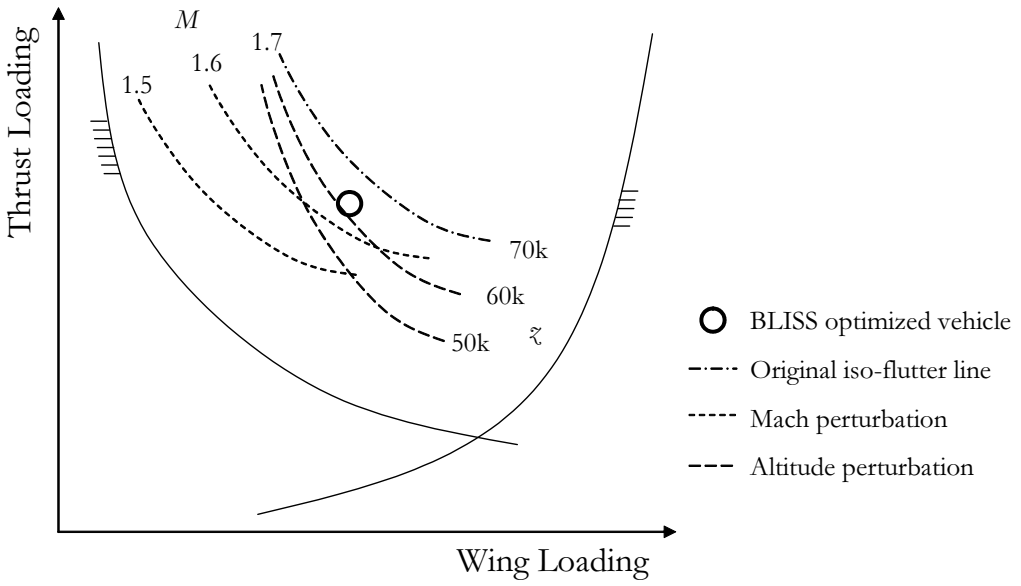

Figure 49: Thrust versus Wing Loading - Iso-aeroelastic Lines

- The BLISS optimization loop changed the geometry input. These are values for sweep, taper, thickness to chord, aspect ratio, etc. These geometric attributes are the lowest level of attributes, and are input to the subsystem (disciplinary) codes.

- The resulting vehicle has *disciplinary metrics* due to its shape and size. These attributes are outputs of the disciplinary codes such as the lift to drag ratio L/D, the drag polar defined by C_{D_0} and C_{D_i}, the take-off weight W_{TO}, etc. By using BLISS, however, these are also called *attributes* since these belong to the subsystem (disciplinary) level and are not global variables.

- The atmospheric/mission conditions are categorized as mission attributes and are the Mach number, altitude, sideslip and angle of attack, or alternatively the number of g's pulled for a maneuver, etc.

- The performance metrics then are the outputs of the BLISS decomposition method as applied to the design problem. These are the system level outputs

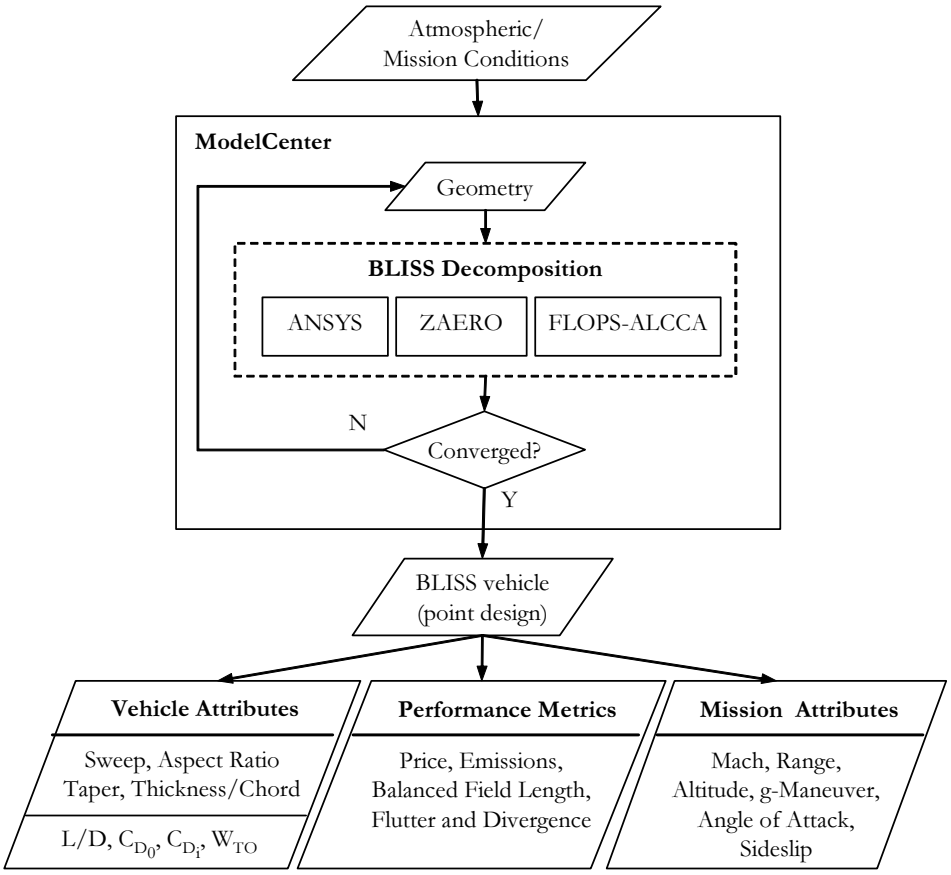

Figure 50: The Method Process Flow

that are of direct interest to the conceptual designer. The list of possibilities includes price, emissions, take-off and landing field lengths, flutter and divergence speeds, etc.

Figure 50 shows the core flow that allows inclusion of aeroelastic constraints in early design stages. As indicated, the core BLISS method only generates one point design. Figure 51 shows that by running a DOE of the atmospheric/mission parameters, the results of these runs can be regressed with respect to the generated DOE table. Using the statistical software JMP [96] to facilitate this regression and visualization, an investigation of trends or sensitivities of attributes with respect to performance metrics can be conducted, or multiple metric contours in a two-dimensional attribute

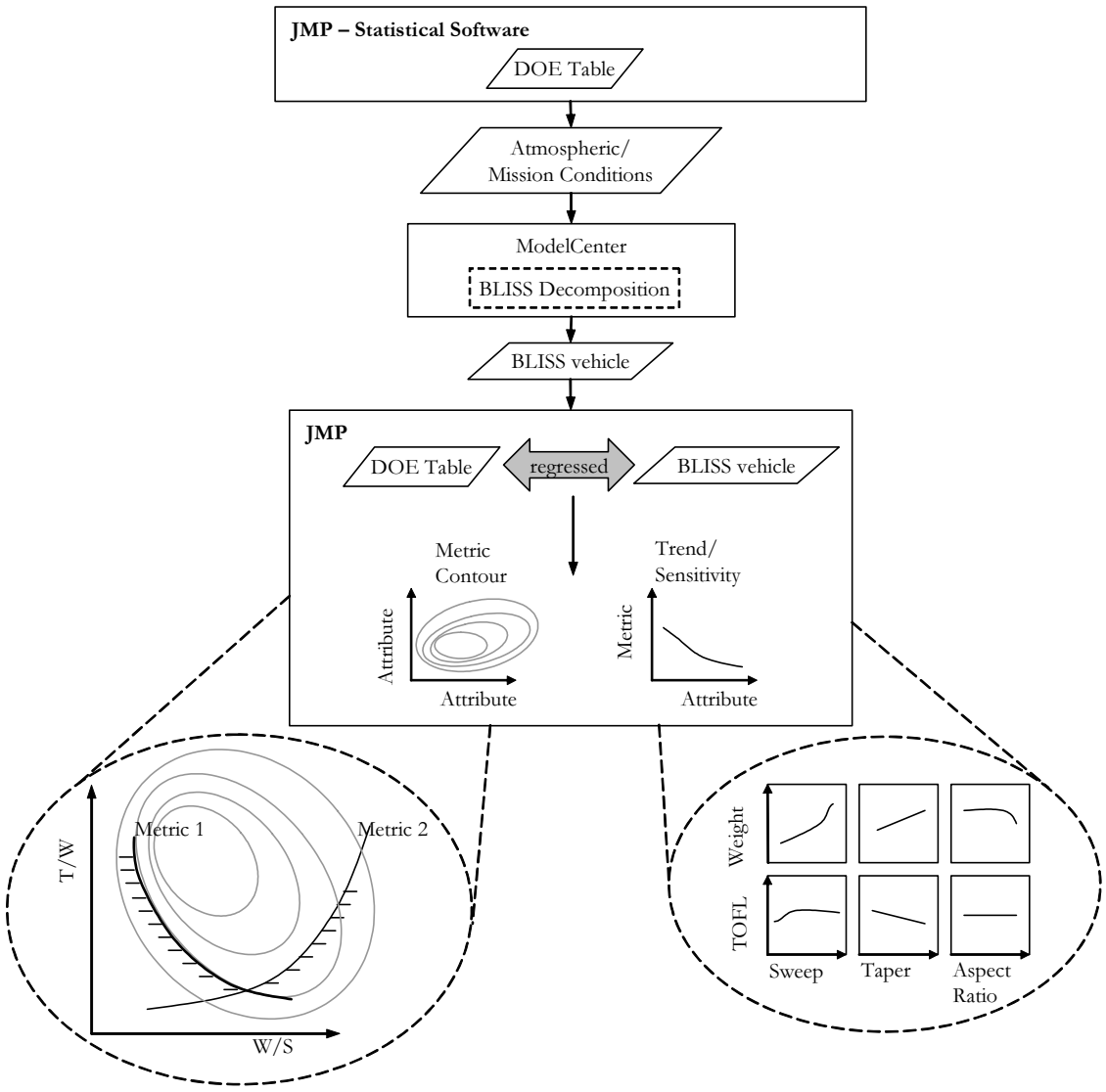

Figure 51: The Possibilities of the BLISS Method Implementation

chart can be plotted. If certain values of a metric are not to be surpassed or have to meet a minimum, the constraints will be isolines of these contours as shown on the bottom left of Figure 51.

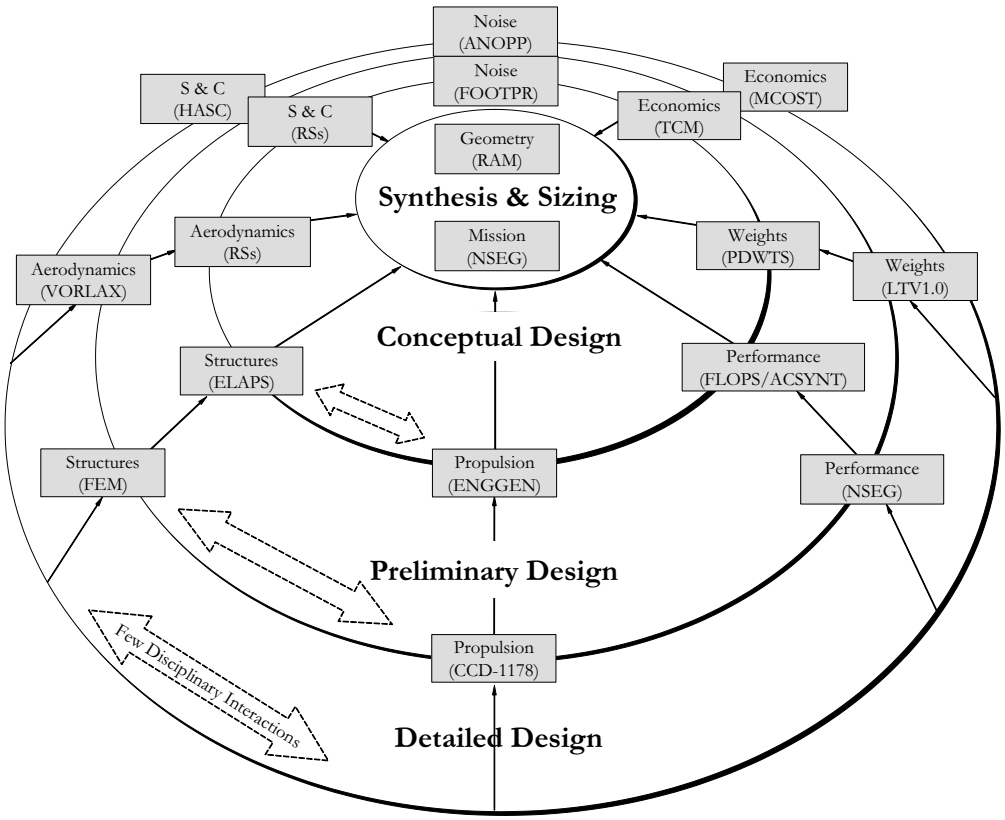

Figure 52: The Integration of Disciplines and Variable Code Fidelity of the Design Process

6.4 Aeroelastic Constraints in the Design Process

The mapping of aeroelastic constraints is part of a general drive to increase the awareness and impact of design decisions at the early design stages. An overview of the different kinds of methods and examples of the increasing higher fidelity codes that can be used for each discipline, is depicted in Figure 52.

At the center is the heart of the design process and includes a mission and a geometry. Every time, with increasing fidelity, the synthesis and sizing is the keystone balancing all the disciplines. A property of this process is the lack of interaction between the various disciplines at all levels.

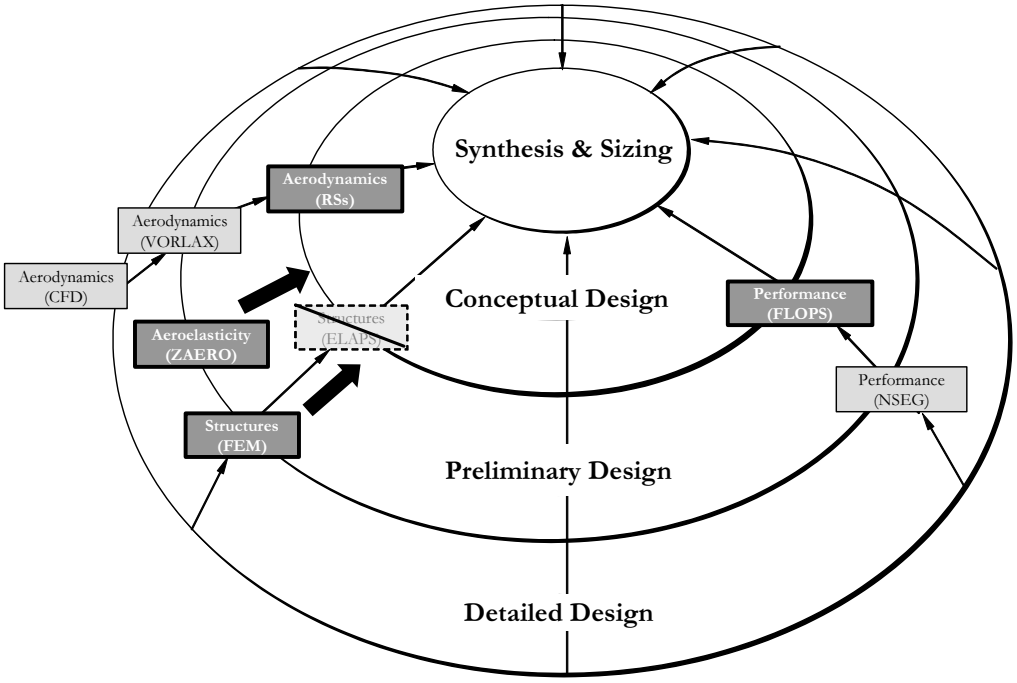

Figure 53: The Impact of BLISS on Integration and Code Fidelity during the Design Process

In a first iteration, at the conceptual design, low-fidelity codes are used, or surrogate models that were constructed off-line for specific ranges. With increasing number of iterations of the design process, a better defined vehicle emerges. Higher fidelity tools can now be used to narrow down which particular vehicle can fulfill the mission while satisfying the disciplinary constraints. Eventually, one vehicle is chosen and analyzed at the detailed design phase with the highest fidelity tools, before the design is sent off to the prototyping and manufacturing floor.

The opening chapters of this document showed the lack of information of aeroelastic constraints at the early stages of design. Given this need, and given the characteristics of aeroelastic information, high-fidelity, physics-based codes were needed to accurately describe these properties. This required that the higher fidelity tools were used at the earlier design stages. Moreover, the aerodynamic and structural discipline coupling had to be included.

Focusing on the aerodynamic, structural and performance disciplines in Figure 53, the value of the method proposed herein is seen. The bold black arrows indicate that the disciplinary codes are moved to the earlier design stage. The aeroelastic calculations are brought forward from the preliminary to the conceptual level, and FE codes replace the lower fidelity structural codes used at the conceptual level until recently [59, 68].

The aerodynamics and synthesis and sizing codes are still relying in the lower fidelity theories in this implementation. The method described is flexible enough though to bring panel method results in the synthesis and sizing code. This aerodynamic research is not new and has been documented extensively in other research [44, 53, 67, 69, 72]. Reduced-order modeling methods are being considered to bring CFD codes to the design stages [26, 49, 88, 126].

6.5 Conclusions

This chapter has dealt with the result strategy used to map the aeroelastically optimized vehicle in the conceptual design graph after the BLISS optimization converges.

This was followed with a description of the overall method to include these aeroelastic constraints as a function of the mission parameters. This also showed that *any* constraint that requires physics-bases codes, can be included in the thrust loading versus wing loading graph with this method. The chapter was rounded off by a situational sketch of this methodology within the larger design process endeavor.

CHAPTER VII

PROOF OF CONCEPT APPLICATION

The proof of concept vehicle was an QSBJ based on existing in-house models. Because of the size of the aeroelastic problem, some assumptions were made in the previous chapters and are listed here again. Especially the aerodynamic drag polars in the synthesis and sizing code require some attention and a validation case is provided, verifying the impact of this assumption. Also, the geometric (global) design variables and structural (local) design variables were screened to reduce computational expense to a manageable size. The focus of the method was on the supersonic regime. However, a subsonic case is also incorporated as a validation of the method.

7.1 Baseline Aircraft

Some characteristics of the two baseline models are provided in Tables 7 and 8. The Figures 7 and 8 show the two vehicles respectively. The mission properties of both vehicles were a 4000 nautical mile design range, and design Mach number of 1.8. A significant difference between the two models, although geometrically almost identical, was the thrust-to-weight ratio or thrust loading. The first vehicle had a value of 0.45, the second one had a much higher value of 0.60. This was mainly due to the following reasons.

- When the second vehicle was investigated, the optimization had a heavier weighing of noise and sonic boom. The objective of the second vehicle was effectively penalized for the noise more then the first vehicle.
- The second vehicle had a higher take-off gross weight because it used a variable sweep outer wing, changing the optimization starting point.

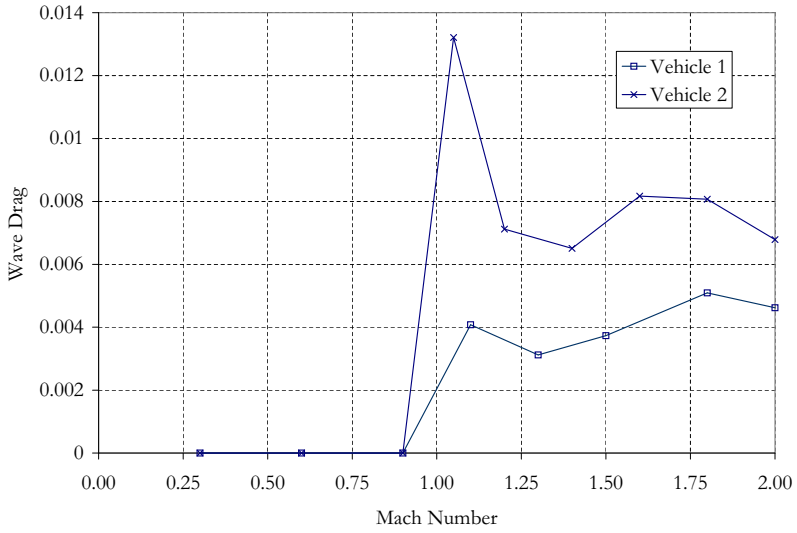

Figure 54: Wave Drag Comparison of Baseline Vehicles

- The second vehicle had a much blunter nose then the first vehicle. This helped control the sonic boom overpressure. The blunt nose however resulted in a drag penalty at cruise. Especially the penetration of the transonic region at the top of climb was the engine's thrust design point. The wave drag for both vehicles can be compared in Figure 54, and shows the higher drag of the second vehicle. This design point resulted in over-sized engines for the take-off case, allowing significant throttle-back on take-off, further helping to reduce take-off noise.

The above reasons combined with the multi-modal, complex design space, resulted in different optimized vehicles, specifically the thrust-to-weight.

The baseline geometry for this research was based on these two existing baselines. A three-view is given in Figure 57. The most significant difference is the choice for a low horizontal tail instead of the T-tail configuration of the two previous vehicles. The fuselage was also a more conventional, cylindrical fuselage, instead of the area-ruled baseline counterparts.

The structural and aerodynamic vehicle model described an entire vehicle as depicted in Figure 58 and 59. The main purpose in doing this resulted from the original

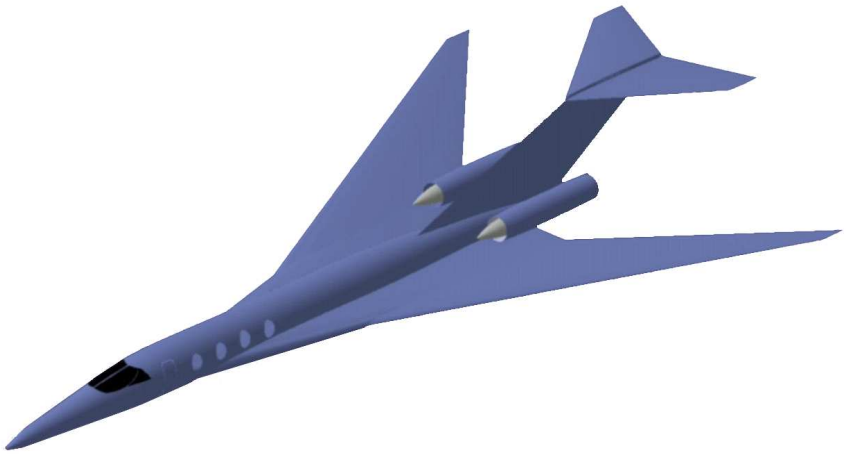

Figure 55: First Baseline Vehicle [67]

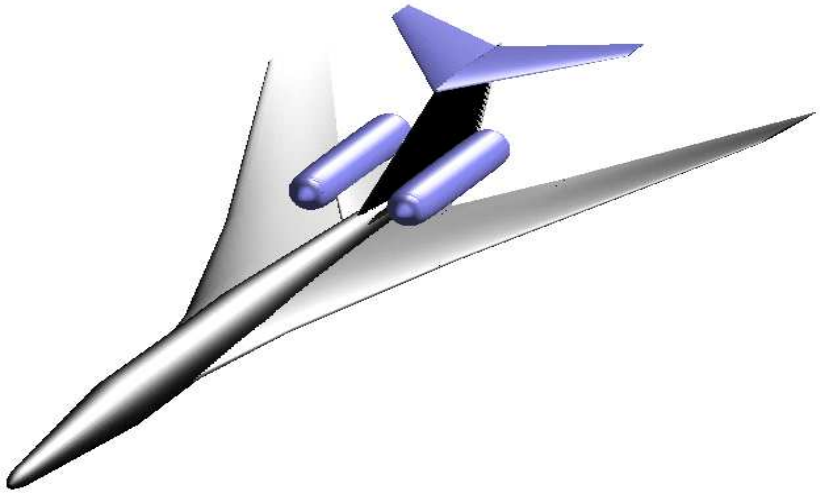

Figure 56: Second Baseline Vehicle [69]

Table 7: Optimized First Baseline Properties [67]

Property	Value	Unit
Fuselage Length	160.0	ft
Fuselage Diameter	8.0	ft
Span	70.7	ft
Wing Sweep	70.0	deg
Planform Area	2500	ft^2
Thickness/Chord	0.025	-
Thrust Loading	0.45	-
Thrust	56421	lb
Wing Loading	50.2	lb/ft^2
Take-Off Gross Weight	125381	lb

Table 8: Optimized Second Baseline Properties [69]

Property	Value	Unit
Fuselage Length	160.0	ft
Fuselage Diameter	7.5	ft
Span	68.0	ft
Sweep	68.0	deg
Wing Area	2268	ft^2
Thickness/Chord	0.025	-
Thrust Loading	0.60	-
Thrust	72395	lb
Wing Loading	53.2	lb/ft^2
Take-Off Gross Weight	120659	lb

desire to be able to include oblique winged concepts.

Another reason lay in the different motions the vehicle can make: symmetric, antisymmetric, and asymmetric motions. A fully modeled vehicle constrained at the center of gravity allows the full scope of motions to be captured with this one boundary condition. In the case only half of a symmetric vehicle was modeled about the longitudinal axis, separate boundary conditions would have to be analyzed to allow for symmetric and antisymmetric motion. This would effectively double the computational effort because the BLISS optimization would have to be repeated separately for each boundary condition. The idea behind the research of analyzing oblique winged

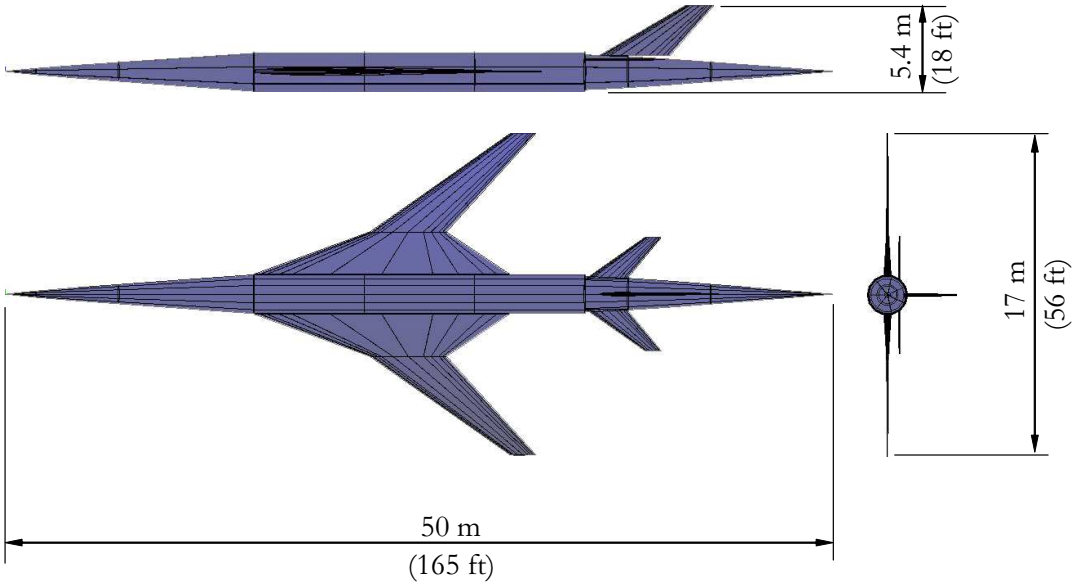

Figure 57: Aeroelastic Research Baseline Vehicle

aircraft also required a fully modeled vehicle and was another motivator.

The computational effort required to solve the structural model, however, is proportional to the number of DOF cubed. By modeling the entire aircraft, the effort is effectively multiplied by eight. This computational effort increase is negligible compared to the boundary condition advantages when using simple models, as is the case in this research.

The model was constrained at the center of gravity to further reduce the computational expense. A free-free aircraft is typically used to perform the aeroelastic analysis. The time savings by omitting the rigid body modes solution was multiplied by the runs in the DOE table and number of RS iterations.

The ANSYS structural model depicted in Figure 58 shows the stick fuselage which uses beams to model the actual real fuselage stiffness. The main wing and horizontal and vertical tail with their associated carry-through structure are also visible. The ZAERO aerodynamic/aeroelastic model is depicted in Figure 59. This shows the cylindrical fuselage, wing and tails made of flat panels.

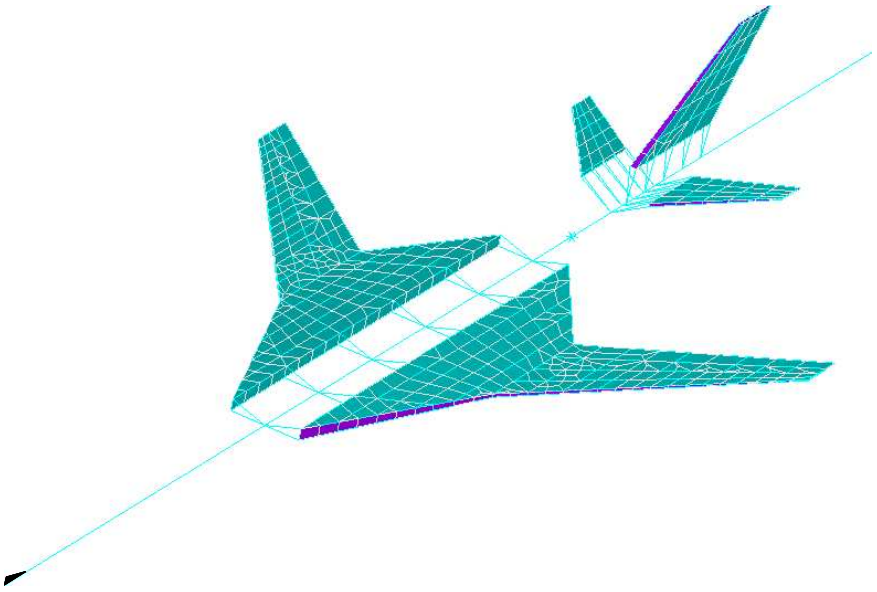

Figure 58: ANSYS Structural Free Mesh

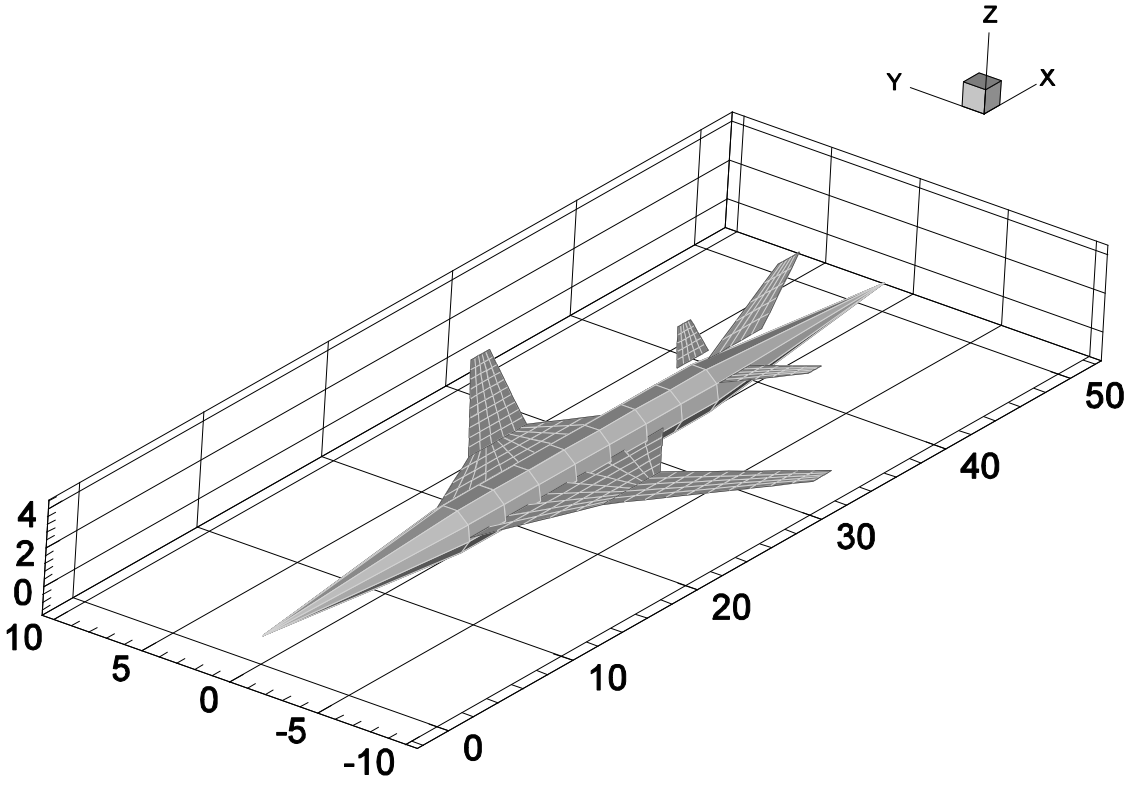

Figure 59: ZAERO Aerodynamic Mesh

7.2 Design Variables

The structural, aerodynamic and performance model were described in previous chapters and are included in Appendix B. The models were all parametric, and the global geometric and local structural design variables are listed in Table 9 and Table 10, respectively. Other design variables include the number of ribs and spars in the wing and tail sections, not mentioned here because these were fixed from the start. The aspect ratio design variable deserves some attention. The definition of it in this research, is the semi-span divided by the root chord. Due to the long root chord, this resulted in a very low aspect ratio design variable. The true aspect ratio of the vehicle is shown in the tables that accompany the results.

If all these geometric variables were included in the DOE/RS creation process, and the local variables in the structural optimization, the result would be an impractical computational effort. The research goal was to prove that the proposed method works, and some variables were omitted to allow this.

Some selection criterion needed to be established in order to logically down-select these design variables. These criteria were chosen to be the sensitivity of the flutter and divergence speed to the specific design variable. Items that had a significant influence on these two properties were selected, while others were fixed and not included in the optimization to reduce computational expense.

7.2.1 Global Design Variable Screening

The screening results of the fifteen global design variables are shown in Table 11 and Figures 97 and 98 in Appendix C.

After careful selection, nine variables were chosen that, respectively, accounted for 62 percent and 74 percent of divergence and flutter speed variability. These variables were chosen such that the main wing had the most design variables but also included some tail variables. Because all three lifting surfaces (wing, horizontal tail,

Table 9: Global Geometric Design Variables

Design Variable	Name	Value	Unit
Main Wing			
Aspect Ratio of Entire Wing	AR	0.55	-
Semi-span of Entire Wing	SPAN	32.0	ft
Kink Location of Wing Joint	KINK	0.30	%
Sweep of Inner Wing	SWEEP1	70.0	deg
Sweep of Outer Wing	SWEEP2	55.0	deg
Taper Ratio of Inner Wing	TAPER1	0.30	-
Taper Ratio of Entire Wing	TAPER	0.10	-
Horizontal Tail			
Aspect Ratio	ARHT	1.0	-
Semi-span	SPANHT	8.25	ft
Sweep	SWEEPHT	55.0	deg
Taper Ratio	TAPERHT	0.40	-
Vertical Tail			
Aspect Ratio	ARVT	0.70	-
Semi-span	SPANVT	10.0	ft
Sweep	SWEEPVT	60.0	deg
Taper Ratio	TAPERVT	0.40	-

and vertical tail) were allowed to flutter and diverge, the variability of these speeds was governed by more then simply the main wing. The final global variables were:

- six for the main wing: span SPAN, aspect ratio AR, the two taper ratios TAPER1 and TAPER, and the two wing sweeps SWEEP1 and SWEEP2;
- two for the horizontal tail: aspect ratio ARHT and sweep SWEEPHT;
- one for the vertical tail: taper ratio TAPERVT.

Some interesting conclusions can be drawn from these results. The sweep of the inner wing SWEEP1 had a significant effect on the geometry at angles higher than 60 degrees. This sweep pushes the outer wing backward and has a significant impact on the flutter speed. Other main wing variables such as SPAN, AR, TAPER1 and TAPER also play an important role in flutter. The sweep of the outer wing portion SWEEP2

Table 10: Local Structural Design Variables

Design Variable	Name	Value	Unit
Main Wing			
Area of Beams in Carry-Through Structure	CT	6.00	ft^2
Area of Beams in Inner Wing	BM1	2.70	ft^2
Thickness of Shells in Inner Wing	SH1	0.25	ft
Area of Beams in Outer Wing	BM2	2.80	ft^2
Thickness of Shells in Outer Wing	SH2	0.16	ft
Horizontal Tail			
Area of Beams in Carry-Through Structure	CTHT	6.00	ft^2
Area of Beams	BMHT	5.00	ft^2
Thickness of Shells	SHHT	0.10	ft
Vertical Tail			
Area of Beams in Carry-Through Structure	CTVT	6.00	ft^2
Area of Beams	BMVT	5.00	ft^2
Thickness of Shells	SHVT	0.30	ft

seems to have a low impact on flutter speed at angles of 55 degrees. However, this variable shows up as of high importance for divergence. Interestingly, if the variables were not important for the flutter speed, these showed up to be important for the divergence speed, and vice versa. This made it difficult to choose a set of variables that had an influence on both properties. This verifies that flutter and divergence speed are contradicting properties with respect to sweep. If sweep is low, divergence variables gain importance and inversely, if sweep is high then flutter variables gain importance.

7.2.2 Local Design Variable Screening

The execution time of the structural optimization was the bottleneck of the BLISS optimization. The runs were performed in parallel, and even with fewer inputs to the DOE than the aerodynamic code, the RS creation for the structural code took the longest to generate as shown in Figure 60.

Keeping the run-time of ANSYS to a minimum was critical to the speed of the BLISS optimization. For the same reason as for the global variables, some variables

Table 11: Global Geometric Design Variables Screening

Design Variable	Normalized Influence	Design Variable	Normalized Influence
Divergence Speed		Flutter Speed	
AR	0.17	SWEEP1	0.23
SWEEP2	0.16	SPANVT	0.18
SWEEPVT	0.13	SPAN	0.11
TAPERHT	0.12	AR	0.10
TAPERVT	0.09	ARVT	0.08
SPANHT	0.09	TAPER1	0.08
ARHT	0.08	TAPERVT	0.08
SWEEPHT	0.05	TAPER	0.08
SPAN	0.04	SWEEP2	0.04
ARVT	0.03	TAPERHT	0.04
TAPER	0.03	SPANHT	0.03
TAPER1	0.01	SWEEPVT	0.01
KINK	0.00	SWEEPHT	0.01
SWEEP1	0.00	ARHT	0.01
SPANVT	0.00	KINK	0.00

were fixed to allow the proposed method to be implemented, while making sure that with increases in computational power, the method could easily handle more variables.

The same screening process was performed as with the global design variables: the local variables that had the most influence on the flutter and divergence speeds were selected, the others were simply fixed or pooled together.

The parametric inner and outer wing had two design variables each, the carry-through structure of the wing, vertical and horizontal tail were another three, and the vertical and horizontal tail each had an additional two variables. A total of eleven design variables thus existed.

Per wing section or tail section, the beam element's cross-sectional area and the shell element's skin thickness could be set. For the carry-through structure the beam's cross-sectional areas were variables. These design variables were listed in Table 10.

The screening results are in Table 12 and Figure 99 and 100 in Appendix C. These

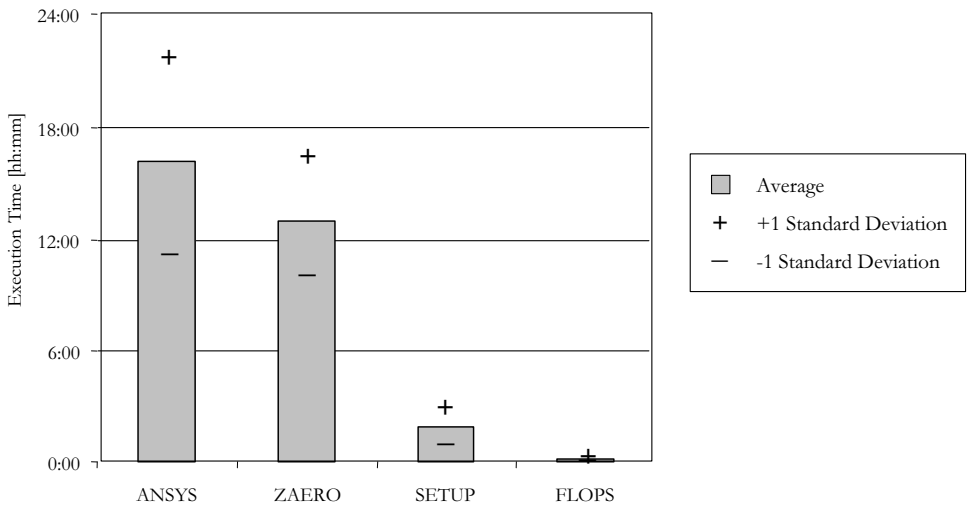

Figure 60: Execution Time Comparison of Different Codes and Setup Time

results highlight the importance of the beam elements. The aeroelastic properties are directly linked to the stiffness of the model. The main contributors to the stiffness are the beam elements, the shell elements do not seem to play a significant role. From an optimization standpoint, the beams are the important elements. Respectively 75 percent and 65 percent of divergence and flutter speed variability can be attributed to `BM1`, `BM2`, `SH2`, and `BMHT`. These were the four design variables in the structural optimization.

7.3 Analysis Assumptions and Modeling Notes

One of the first goals of this research was the implementation of BLISS decomposition of the aeroelastic problem. To reduce the initial size of the problem some assumptions were made in the implementation as described in Chapter 5. These and others are listed here because of their impact and influence on the results.

7.3.1 Analysis Assumptions

So far, three main assumptions were made that simplified the problem to a tractable size. Some of these were already touched upon in the previous chapter.

Table 12: Local Structural Design Variables Screening

Design Variable	Normalized Influence	Design Variable	Normalized Influence
Divergence Speed		Flutter Speed	
BMHT	0.32	BM1	0.36
BM1	0.31	BM2	0.14
CTHT	0.10	CTVT	0.13
BMVT	0.08	SH2	0.09
CT	0.07	SH1	0.09
SH2	0.07	SHVT	0.06
BM2	0.05	BMHT	0.05
SH1	0.00	BMVT	0.03
CTVT	0.00	CT	0.02
SHHT	0.00	CTHT	0.02
SHVT	0.00	SHHT	0.01

1. The vehicle structure in ANSYS was optimized with neither airloads nor fuel. This already resulted in the shell elements of the wing not being of much importance in determining the stiffness of the model.

2. The aerodynamic data used to calculate the performance of the vehicle are based on the baseline vehicles. This resulted in significant wing changes from the baseline vehicle's wing. The aerodynamic penalty or improvement would not be captured by the synthesis and sizing code.

3. For simplification of the structural and aerodynamic models and in order to reduce the number of design variables, no control surfaces were considered, this circumventing a trim analysis. The resulting weight of FLOPS-ALCCA was not fed back to ZAERO for this reason. This meant that the angle of attack was a parameter.

To include fuel and airloads, a more accurate drag polars, and a trim analysis would entail a significant effort. Combined with the limited computational resources, this was not done. The initial research questions to be answered were to prove the

feasibility of BLISS decomposition to generate aeroelastic constraints in a design environment with large data flows. The initial price paid were the above three main assumptions.

7.3.2 Model and Optimization Notes

A few important topics are highlighted that occurred commonly throughout the RS generation phase.

7.3.2.1 Design Variables

Several internal code and physical constraints prevented certain settings of variables. Sometimes, these constraints prevented variables from obeying the optimization wants.

- The difference between `SWEEP1` and `SWEEP2` was never allowed to be greater then five degrees.
- `SWEEP1` and `SWEEP2` were not allowed to be less then 50 degrees since that would put the wings too much in front of the shock wave cone.
- The sweep of the horizontal tail `SWEEPHT` and aspect ratio `ARHT` were not allowed over 65 degrees and one respectively. Allowing more resulted in impractical tails.
- Sometimes the synthesis and sizing code could not fly the mission with a certain vehicle. After many runs, general observation showed that FLOPS-ALCCA had some variables that were critical for flying the mission. Internal code constraints led to the following general observations: taper of the vertical tail `TAPERVT` could not be set below 0.26, semi-span of the wing `SEMISPAN` could not be set below 7.00, and wing aspect ratio `AR` could not be set below 0.45.
- Other settings preventing the synthesis and sizing code from flying the mission were the thrust-to-weight setting. Due to the drag polars of the second baseline

vehicle being used, a high value of 0.75 was needed to obtain robust results from this code.

- The structural code became unstable when the taper ratio of the wing `TAPER` was chosen to be less then seven percent. This was caused by the number of chosen spars, and resulting bad aspect ratio elements in the tip region if the taper ration was too small.

These self-imposed problem limitations were mainly attributable to the lack of:

- airloads on the horizontal tail structure;
- updated aerodynamics penalizing sweeps forward of fifty degrees for high speed performance and the delta planform for low speed performance.

One could note that these constraints require a *human in the loop*. More important though is the argument that more disciplines and coupling ties need to be implemented in this problem. It would not be uncommon that inclusion of stability and control, better aerodynamic drag prediction, structural sizing including loads after a trim analysis, and more performance constraints such as landing and take-off would solve the current imposing of artificial constraints on these variables.

7.3.2.2 Response Surface Validation

Validating RS equations is an important step. After generating RS equations from the collected data, one typically validates these. This validation step tests the equations at random points within the range of validity to verify their accuracy. The number of test points usually is equal to the number of design points used in the DOE.

This proved to be problematic. Because the generation of RS equations is a key part of BLISS, the requirement to validate each newly created RS would be a significant computational burden for BLISS, especially for this aeroelastic problem which uses time expensive physics-based codes.

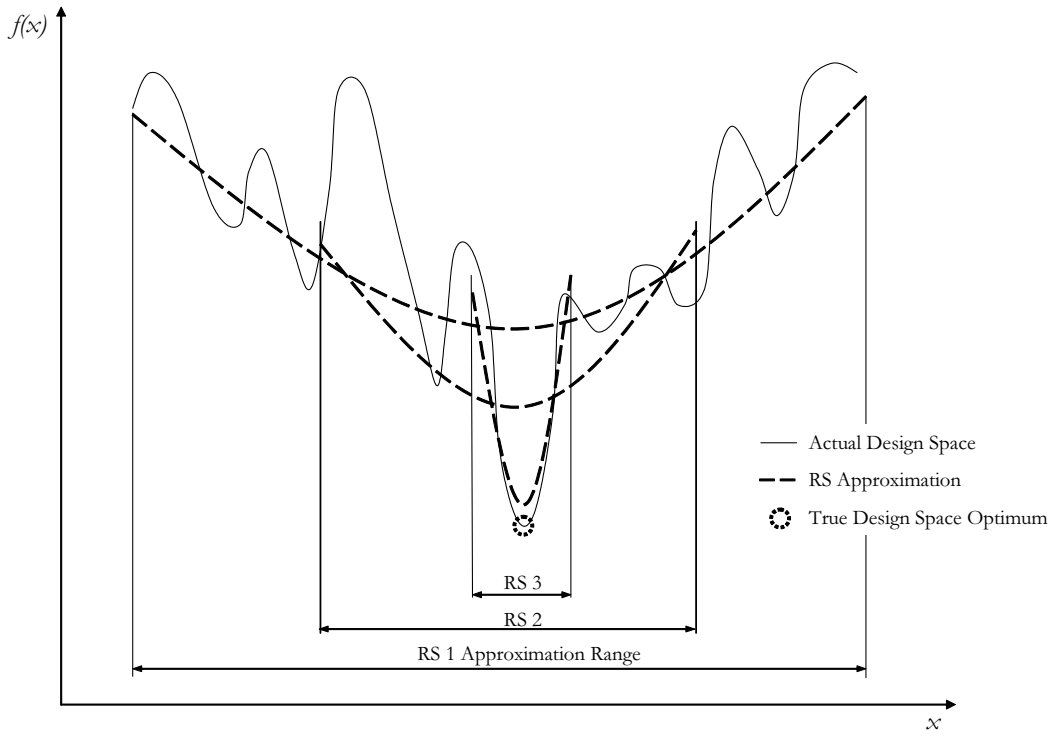

Figure 61: Response Surface Approximation

From the graphs of predicted flutter and divergence speed in Appendix D, the RS equations were not accurately approximating the design space in the first iterations of the problem. This was due to the chosen wide ranges. After a few iterations, more narrow ranges were chosen and certain variables were fixed as these kept hitting internal code constraints. A good example of these internal constraints is shown by the semi-span `SPAN` and aspect ratio `AR` design variables. Fixing the value of certain variables and reducing the width of the ranges helped in reducing the prediction error of the RS equations.

When it was determined that BLISS had converged, the final design point obtained with the RS equations was verified with a one-shot check-run execution of the codes. The predicted RS values were updated with these actual check-run results. This procedure bypassed the requirement to validate the RS equations at each new iteration step of BLISS, and reduced the computational expense significantly.

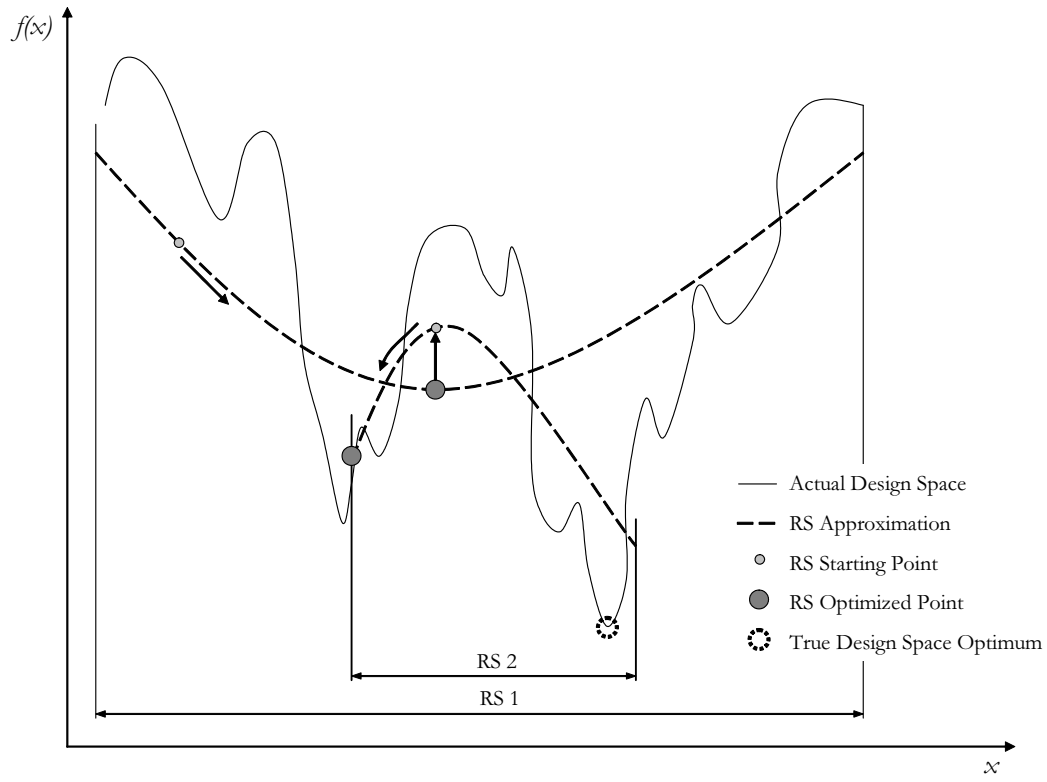

Figure 62: Response Surface Erroneous Approximation

A problem nevertheless persists with RS approximation of this multi-modal space. Consider Figure 61, where an initial RS approximates a large region. By subsequent narrowing of the ranges and generating a new RS equation of the design space, one finds the optimum.

Figure 62 shows that convergence to a non-optimum point in the design space is possible. The validation requirement would not solve this erroneous optimization. A radically different approximation technique is needed. One of the items in the future work is the use of surrogate models that allow capture of some of the multi-modal aspects of the design space.

Another reason which may contribute to the poor modeling of the quadratic models may be discontinuous flutter modes. Bisplinghoff [16] and Fung [31] portray most aeroelastic trends as smooth, so that quadratic models may capture trends accurately.

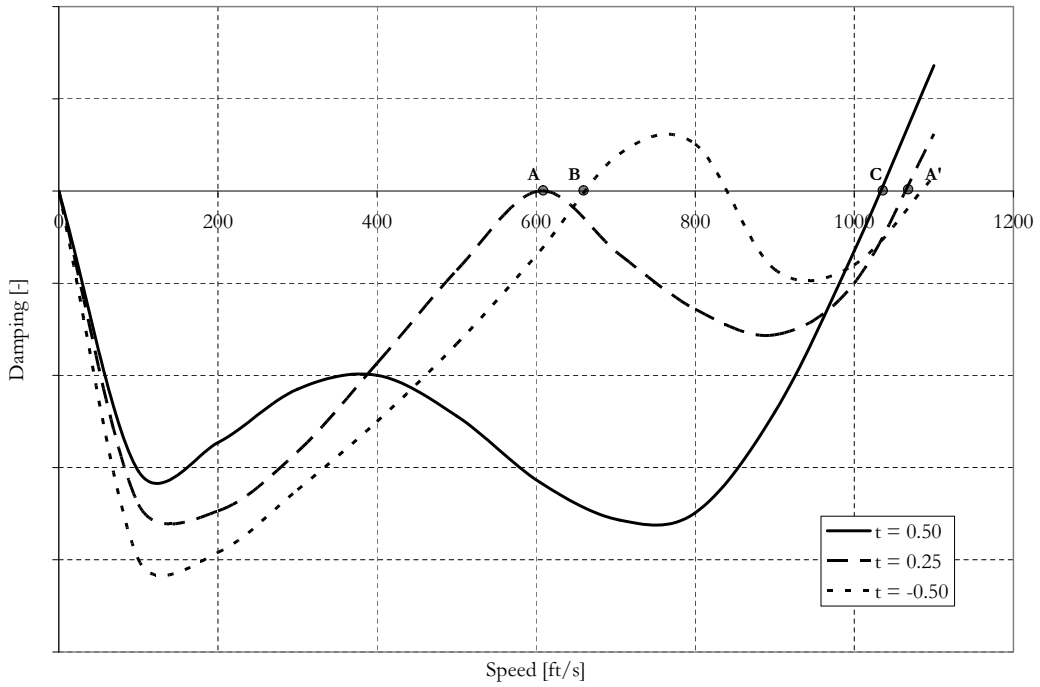

Figure 63: Example of Hump Flutter Modes

There is a problem with flutter hump modes, however, and results seem to indicate the presence of this problem. Particularly the unsteady nature of the predicted flutter and divergence speeds graphs in Appendix D was troubling.

RS equations captured the influence of various variables on the aeroelastic properties, such that a critical constraint can be introduced in the dynamic design space. A smooth change of properties with respect to design variables is usually assumed when performing this curve fitting. Assume a few flutter modes are obtained for different settings of a design variable, then a curve-fitting flutter speed would work. The RS equation would approximate the critical flutter boundary, not knowing the underlying flutter modes.

In aeroelastic optimization one should expect that as the design variables change, the flutter modes may switch. From Figure 63 follows that increasing a notional non-dimensionalized design variable t from 0.50 to -0.50, the hump mode is forced to change from the solid line, to the dashed line, and eventually to the dotted line. The

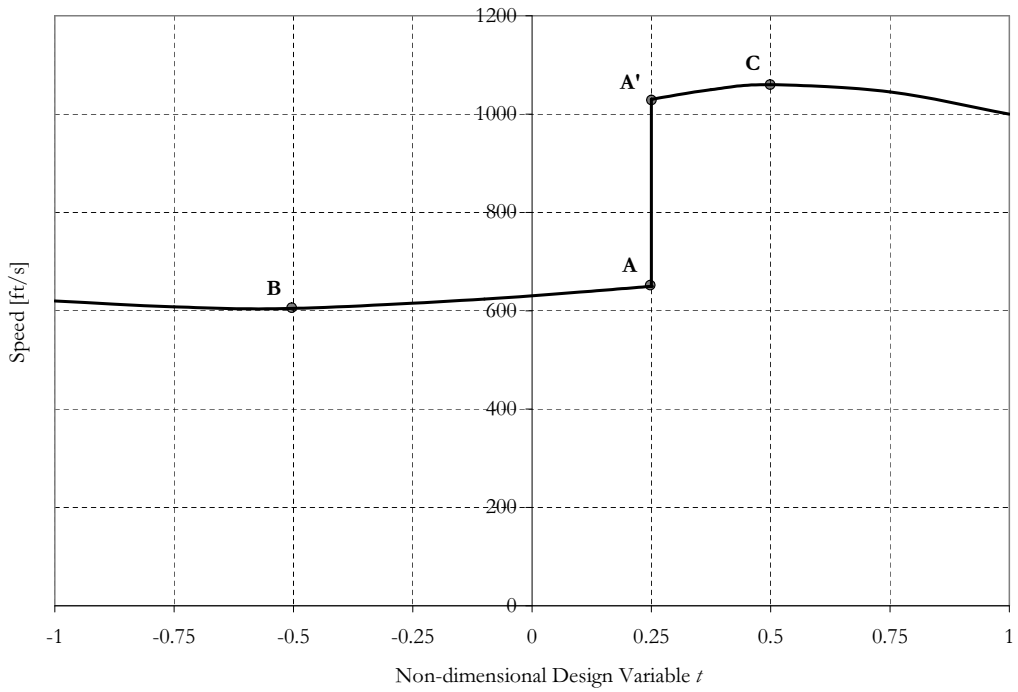

Figure 64: Example of Flutter Speed Value Discontinuity

initial flutter value was at point C and as the design variable setting decreases to 0.25, there are two possible flutter points: A' and A. As the design variable is further decreased beyond 0.25, the speed increases from A to B.

Theoretically, mode i has a flutter speed V_i and mode j has V_j, such that $V_i < V_j$, so that V_i governs. Also assume that after the design variable change $V_j < V_i$ and V_j governs. This makes the flutter constraint written as $g = 1 - \frac{V_i}{V_{limit}} \leq 0$ slope-discontinuous, or even value-discontinuous with respect to the design variable as depicted in Figure 64. This potentially ruins the prospect of representing the flutter constraint by RS since sampling errors are introduced as in Figure 65.

The remedy is to go with not one but a few flutter constraints for the lowest flutter modes, one constraint per such mode. Since this behavior of hump modes was observed in the large flexible Boeing's SST with a similar double delta wing, the occurrence of hump modes is expected and it is hypothesized that these exist here [37, 91].

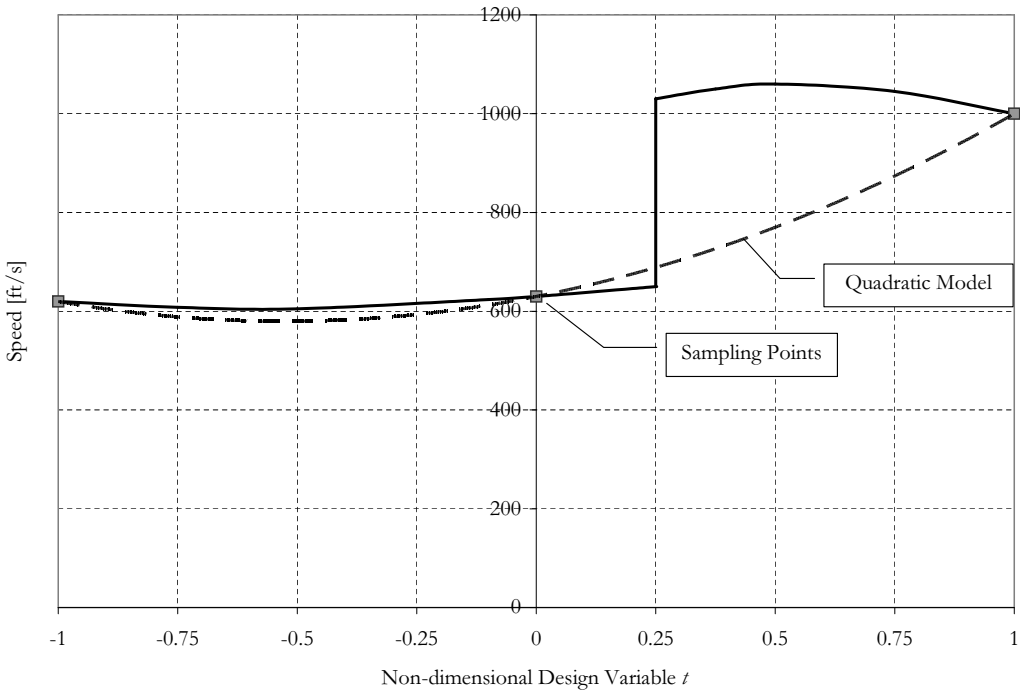

Figure 65: Example of Flutter Sampling Errors

7.3.2.3 Optimization and Convergence

Determining when BLISS had converged was not trivial because in the mathematical sense of the word this did not happen. The graphs to be shown in Appendix D indicate asymptotic convergence to a point and a solution that had optimized to within a few percents.

Full convergence in the mathematical sense would require many more runs. A possible reason for this is due to the weak coupling present between the structural and aerodynamic RS. This coupling can only be made stronger if the mesh in the structures code is changed to a mapped mesh, which fixes the number of nodes and node locations in the structural model and facilitates the addition of more feedback to the aerodynamics code. However, using a mapped mesh incurs a penalty in terms of the amount of perturbation possible of the structural model. This is a trade-off.

Determining the convergence of BLISS was an example of human interaction and

the requirement for human insight. This is not uncommon in design, especially at the conceptual design stage. For example, when a new airplane is envisioned, manufacturers and airlines come together to determine what increases in fuel efficiency are needed to make the airplane marketable. When goals are determined, experts of both fields determine what ought to be achievable and what ought to be a minimum requirement. These discussion are based off of market and conceptual sizing studies. In essence, an optimum is found between the investment requirement and the potential profit for airline and manufacturers.

The same is true for BLISS at the conceptual level. The designer determines what amount of computational time is to be invested to find the optimum point, and at which point the added data points from a new run will no longer influence the design significantly. As such there were no rigorous stopping criteria; rather, good judgment was used.

The person in the loop is also needed to increase the speed of convergence. For example, the system optimization tried to push the design variables into areas that were prevented by aforementioned internal code constraints. In this case, when an indication was present of bumping into such a constraint consecutively, human insight decided that these variables should be fixed. This reduced the number of runs in the DOE table, sped up the RS creation phase, and reduced the overall optimization complexity.

These problems could be earmarked as detriments to BLISS, and that such problems and restrictions should be anticipated. Safeguards could be put in place such that the system optimization adjusts its search algorithm. However, to implement such checks would require unwieldy logic. It is also futile, as these could never be made general enough for each vehicle design problem. In addition, the design process for any vehicle always requires experts in the field, i.e. persons in the loop, to determine if the computer codes are generating technically feasible and physically viable

solutions.

7.4 Validation and Checks

Before applying the BLISS method and obtaining results, a few last checks were needed. An assessment of the impact of mesh density and number of eigenmodes was performed. Results of this section are compiled also in Appendix C.

7.4.1 Aerodynamic and Structural Mesh Density

The effect of meshes and modeling on aeroelasticity has been investigated in the past. Striz and Venkayya [108] made some conclusions as to the effect of the spanwise and chordwise structural and aerodynamic mesh densities on aeroelastic properties.

From a structural and aerodynamic point of view, a coarse mesh resulted in a conservative estimate of the flutter speed. This also had the added benefit of reducing computational expense. Overall, the results from a coarse model did not differ much from a more detailed model.

Figures 101 to 104 in Appendix C show which mesh variables had the most influence on the aeroelastic properties. The only aerodynamic mesh variable that had an influence was the chordwise outer wing `CHD2` on the flutter speed. The other aerodynamic mesh variables had little or no influence on flutter or divergence speed as shown in Figures 101 and 102. Structurally some more mesh variables seemed to play a role. From a divergence speed perspective, the beam elements in the horizontal tail `BMHT` and shell elements in the inner wing `SH1` played a role as shown in Figure 103. For the flutter speed, shell elements in inner and outer wing `SH1` and `SH2`, and beam elements in outer wing `BM2` had the most influence as shown in Figure 104.

The mesh variables were set using the pointers from Striz and Venkayya: structurally the chordwise amount of elements needs more attention than the spanwise, and the opposite is true of the aerodynamic model. Structurally there was no distinction between chordwise and spanwise distribution: only the element lengths of the beam

Table 13: Structural Mesh Variables

Mesh Variable	Name	Maximum Length	Unit
Main Wing			
Beam Elements for Carry-Through	`CT`	1.0	ft
Beam Elements for Inner Wing	`BM1`	1.0	ft
Shell Elements for Inner Wing	`SH1`	2.5	ft
Beam Elements for Outer Wing	`BM2`	1.0	ft
Shell Elements for Outer Wing	`SH2`	2.5	ft
Horizontal Tail			
Beam Elements	`BMHT`	1.5	ft
Shell Elements	`SHHT`	1.5	ft
Vertical Tail			
Beam Elements	`BMVT`	1.5	ft
Shell Elements	`SHVT`	1.5	ft

and shells could be set. These are summarized in Table 13. The aerodynamic mesh is summarized in Table 14. Here, next to the pointers from Striz and Venkayya, the root chord length of the inner wing was taken into consideration and more elements were added chordwise.

7.4.2 Effect and Correlation of Eigenmodes

7.4.2.1 Sensitivity to Eigenmodes

The number of eigenmodes used to determine the flutter speed is typically on the order of 50. A study was conducted to verify if the flutter or divergence speed varied significantly with number of modes.

The range of number of eigenmodes used to calculate flutter varied from 55 to 70 in steps of five. This sweep showed no significant differences between flutter speeds as shown in Figure 105 in Appendix C. Interestingly there is a certain spread of flutter speeds. However there is no significant increase in correlation between flutter speeds with increasing number of modes. This is perhaps indicative of the nonlinear, complex nature of flutter.

Table 14: Aerodynamic Mesh Variables

Mesh Variable	Name	Number of Elements	Unit
Main Wing			
Spanwise Elements for Inner Wing	SP1	6	-
Chordwise Elements for Inner Wing	CHD1	11	-
Spanwise Elements for Outer Wing	SP2	11	-
Chordwise Elements for Outer Wing	CHD2	6	-
Horizontal Tail			
Spanwise Elements	HT	6	-
Chordwise Elements	HT	6	-
Vertical Tail			
Spanwise Elements	VT	6	-
Chordwise Elements	VT	6	-

The same correlation plot is shown for divergence speed in Figure 106 in Appendix C. There is a very high correlation independent of number of modes used.

7.4.2.2 Eigenmode Correlation

An interesting property noted in Chapter 5 was the pooling of eigenvalues and the property that all modes are important to determine the flutter speed but not all are important to the variability of flutter speed.

The first property is illustrated with Figure 108. Two distinct groups are highlighted on this figure. The eigenvalues 29 and above all behaved in a similar fashion and had a correlation mean and standard deviation of (0.88,0.06). The lowest value was 0.62. The second group was comprised of the eigenvalues 18 to 28. A similar pattern here existed with correlations mean and standard deviation of (0.77,0.07) with the lowest value in the group 0.61. A third group consisted of the lowest seventeen modes as shown in Figure 107. Similar patterns could be found but the groups of eigenmodes were smaller.

In all seven groups, one *pooling* parameter determined the behavior of the grouped eigenmodes well. This property allowed the eigenfrequencies to be grouped: one

master pooling factor determining the trend of the remaining slaved eigenvalues.

The second property is the effect eigenmodes have on the flutter speed. The observations of an individual perturbation test of all the eigenvalues are listed in Figures 109.

- The first two eigenvalues had the biggest effect on flutter speed. These contributed to over 50 percent of the variability of the flutter speed.
- Some higher-order eigenfrequencies are important. However, most of the variability is attributable to the lowest twenty eigenfrequencies. This verified the premise that the critical flutter modes are usually due to the coupling of the lowest order structural modes. This is also the basis for the modal modeling simplification mentioned in Chapter 4, that the structural deformation can be sufficiently represented by superposition of lower order modes.

The divergence speed is usually also affected by the number of modes. The static behavior of a wing is gleaned from (dynamic) modes, and generally more modes are even needed to correctly assess the static properties of the vehicle. Figure 110 shows which eigenvalues influence the divergence speed variability the most.

7.5 Mapping to the Design Space

An important methodological contribution to the aeroelastic community is the connection of traditional designers practices with multidisciplinary research using BLISS. The method to do this was discussed in the implementation chapter. Free parameters left in the model are Mach number M, altitude z, and the angle of attack α. The settings were determined by a DOE which captured the main effects of these three variables. The four runs are shown in Table 15. These runs are entirely theoretical to capture the main effect of these parameters on flutter and divergence speed; the vehicle would never fly at Mach 2 at an altitude of 10000ft. These theoretical runs

Table 15: BLISS Design of Experiments Runs

	M [-]	α [deg]	z [ft]
First Case	2.0	2.0	10000
Second Case	2.0	-2.0	60000
Third Case	1.5	2.0	60000
Fourth Case	1.5	-2.0	10000

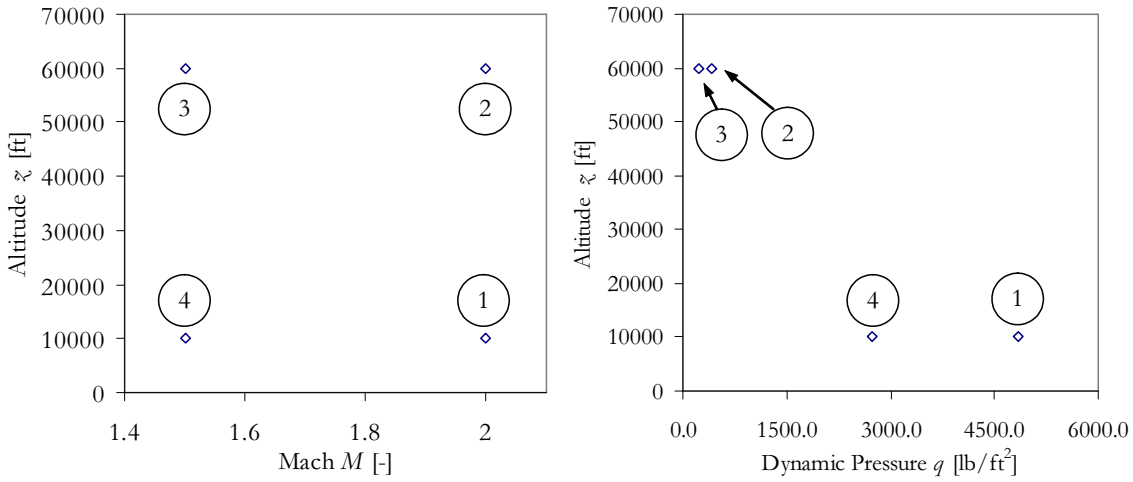

Figure 66: Converting the Design Space from Mach to Dynamic Pressure

simply sample the extremities of the design space, as a DOE is designed to do, and infers information from those runs.

The choice of Mach number as a parameter was not a good one. When plotting the results as a function of dynamic pressure q, the speed was squared and resulted in a severely skewed design space from the dynamic pressure perspective as illustrated in Figure 66.

Due to the impractical corner point cases of the DOE and the choice of Mach as the free parameter instead of dynamic pressure, high values of dynamic pressure in certain instances were attained.

7.5.1 First Optimization Run and Scaling

The BLISS optimization results can be found in Figures 111 through 119 in Appendix D. The gray outlines that are drawn in the graphs with geometric variables, objective function weight w, and eigenvalue factors are the validity ranges of the RS equations.

A plot of the change in planform from the starting point to the optimized one is portrayed in Figure 67 and a summary of the optimized planform characteristics is listed in Table 16.

Subsequent projection of the resulting design in the thrust loading versus wing loading graph resulted in Figure 68. The iso-flutter and iso-divergence lines were obtained by scaling the converged BLISS vehicle up and down by ten percent in wing area S and thrust T. These eight points were run with a one-shot code execution. This figure also shows the effect of changing the flutter and divergence constraint. The plotted constraints are for dynamic pressure q of 1000, 2000 and 3000 lb/ft^2, and the arrow indicates an increasing constraint value.

The white space not covered by either hashed out green or red surfaces is feasible design space where the flutter and divergence constraints are met. If higher values for flutter and divergence constraint are required, this white design space shrinks. In effect, higher constraint values reduce the design space of the designer.

Figure 68 is only a snapshot of the design space. There are many more constraints that govern the converged airplane. Figure 69 shows additional constraints that can be plotted such as emission, price, and take-off field length. From an aeroelastic viewpoint though, both flutter and divergence are bounding the design space.

7.5.2 Second Optimization Run

The BLISS optimization results can be found in Figures 120 through 128 in Appendix D. A plot of the change in planform from the starting point to the optimized one is portrayed in Figure 70 and a summary of the optimized planform characteristics is

listed in Table 17.

Subsequent projection and scaling in a thrust loading versus wing loading graph resulted in Figure 71 and a scaling of flutter and divergence speeds again for dynamic pressures of 1000, 2000, and 3000 lb/ft^2. Additional constraints were plotted in Figure 72. It is interesting to note that the divergence speed is not in the design space plots except for a small indication of its existence at the top. Here only the flutter speed limits the design space from an aeroelastic perspective.

7.5.3 Third Optimization Run

The BLISS optimization results can be found in Figures 129 through 137 in Appendix D. A plot of the change in planform from the starting point to the optimized one is portrayed in Figure 73 and a summary of the optimized planform characteristics is listed in Table 18.

Subsequent projection and scaling in a thrust loading versus wing loading graph resulted in Figure 74 and with additional constraints in Figure 75. The flutter speed is the only constraint drawn on these plots, the divergence speed constraint is always met for this case.

7.5.4 Fourth Optimization Run

The BLISS optimization results can be found in Figures 138 through 146 in Appendix D. A plot of the change in planform from the starting point to the optimized one is portrayed in Figure 76 and a summary of the optimized planform characteristics is listed in Table 19. Subsequent projection and scaling in a thrust loading versus wing loading graph resulted in Figure 77. Again as with the first case, both flutter and divergence are potential limits in the design space.

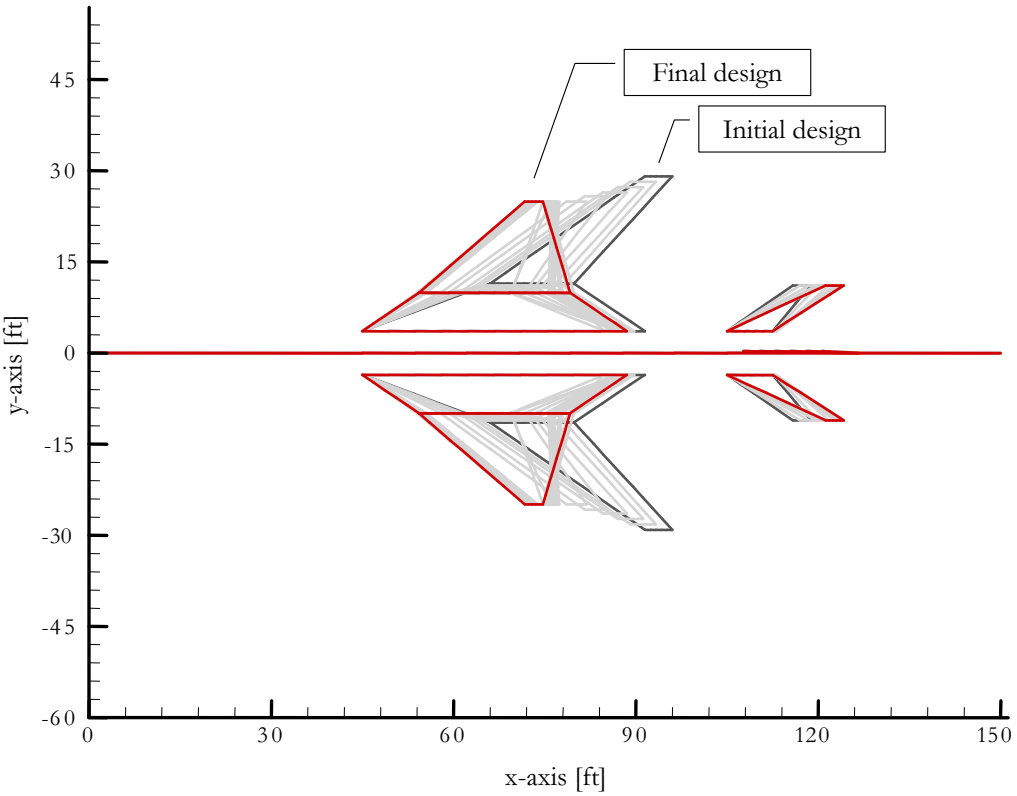

Figure 67: Optimization of First DOE Case

Table 16: Optimized Properties for First Case

Property	Start Value	Optimized Value	Unit
Span	64.0	54.6	ft
Aspect Ratio Entire Wing	2.95	1.27	-
Taper Inner Wing	0.30	0.57	-
Taper Entire Wing	0.10	0.07	-
Sweep Inner Wing	70.0	55.4	deg
Sweep Outer Wing	55.0	50.0	deg
Theoretical Wing Area	1393	2351	ft^2
Thrust Loading	0.75	0.75	-
Thrust	76608	65537	lb
Wing Loading	73.3	37.2	lb/ft^2
Take-Off Gross Weight	102144	87383	lb

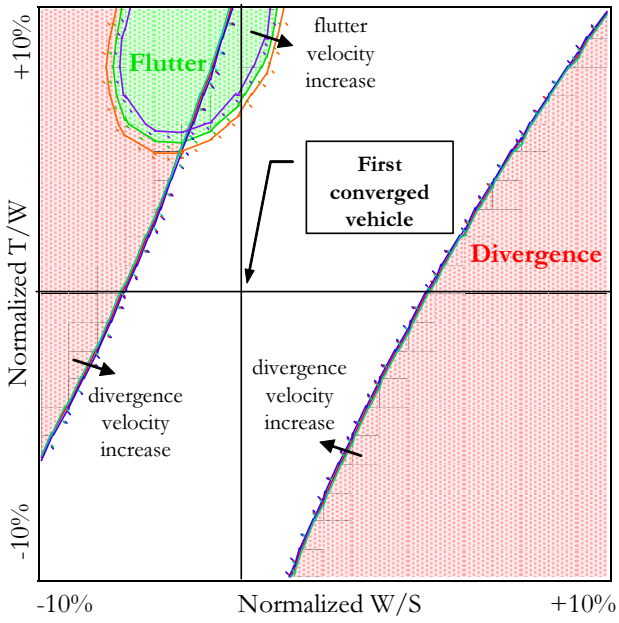

Figure 68: Iso-flutter and Divergence Lines for First Vehicle

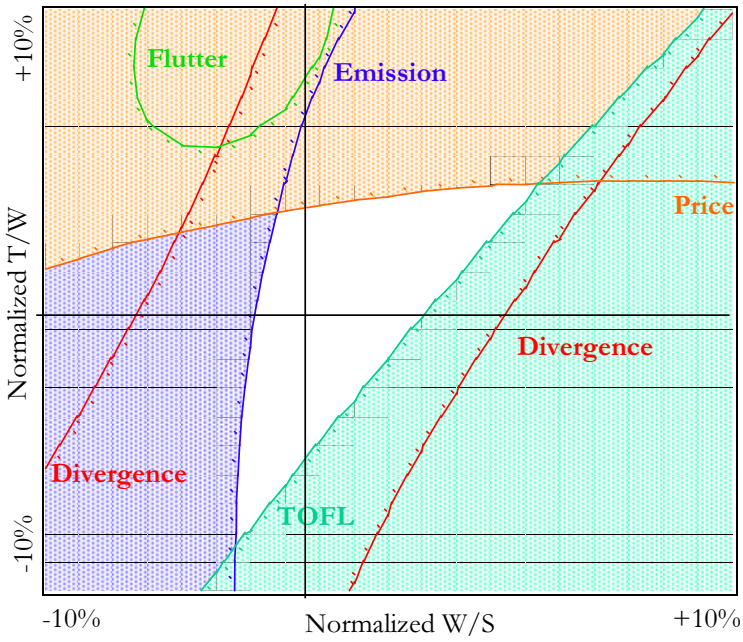

Figure 69: Additional Constraints in the Design Space for First Vehicle

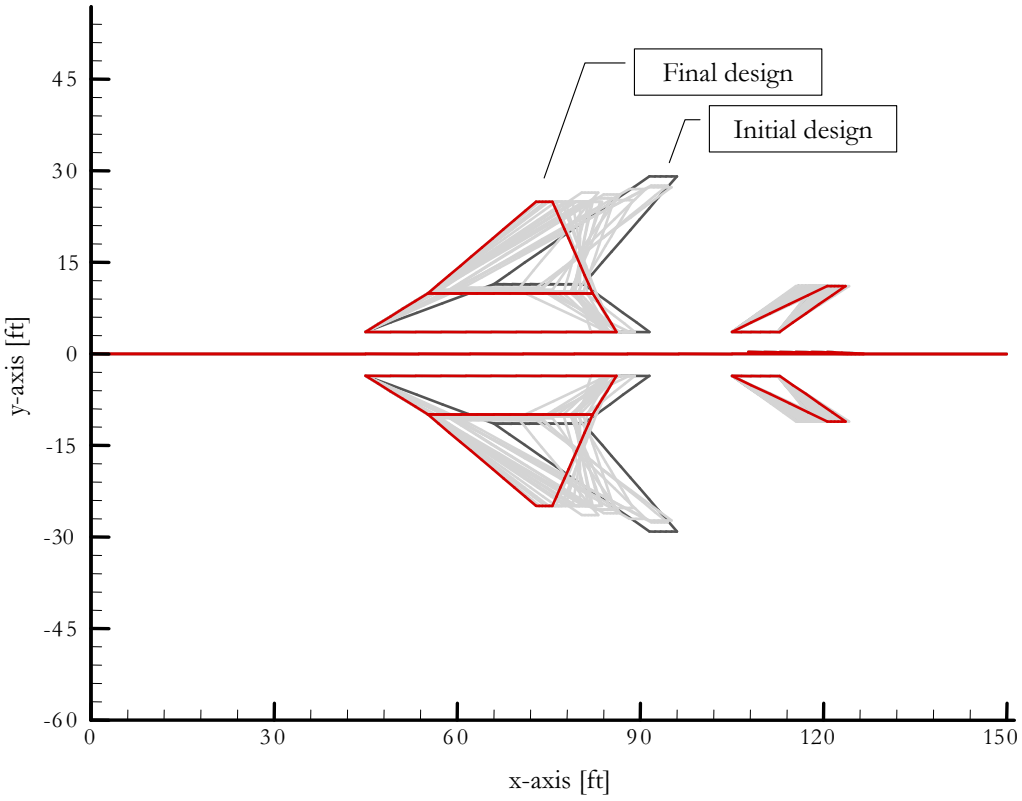

Figure 70: Optimization of Second DOE Case

Table 17: Optimized Properties for Second Case

Property	Start Value	Optimized Value	Unit
Span	64.0	54.6	ft
Aspect Ratio Entire Wing	2.95	1.15	-
Taper Inner Wing	0.30	0.66	-
Taper Entire Wing	0.10	0.07	-
Sweep Inner Wing	70.0	58.0	deg
Sweep Outer Wing	55.0	50.0	deg
Theoretical Wing Area	1393	2584	ft^2
Thrust Loading	0.75	0.75	-
Thrust	75902	65466	lb
Wing Loading	72.7	33.8	lb/ft^2
Take-Off Gross Weight	101202	87288	lb

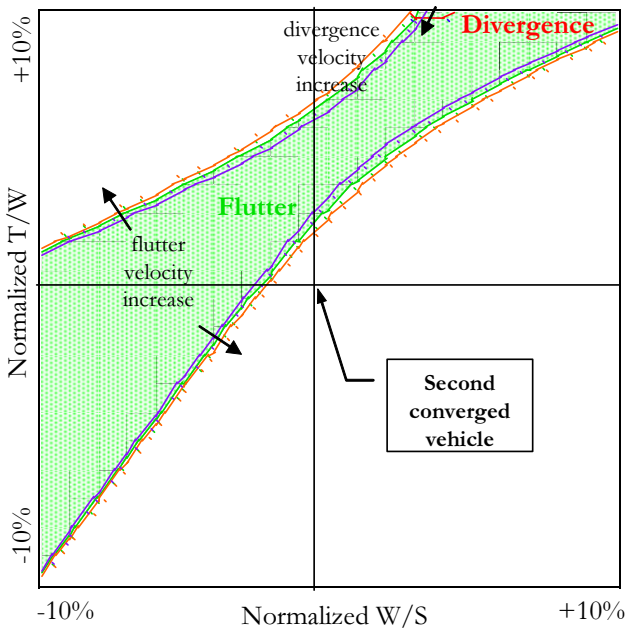

Figure 71: Iso-flutter and Divergence Lines for Second Vehicle

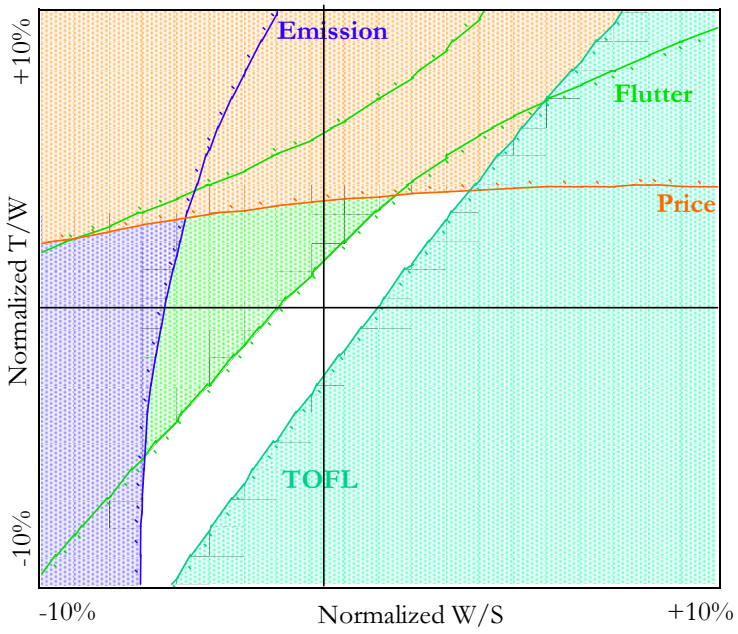

Figure 72: Additional Constraints in the Design Space for Second Vehicle

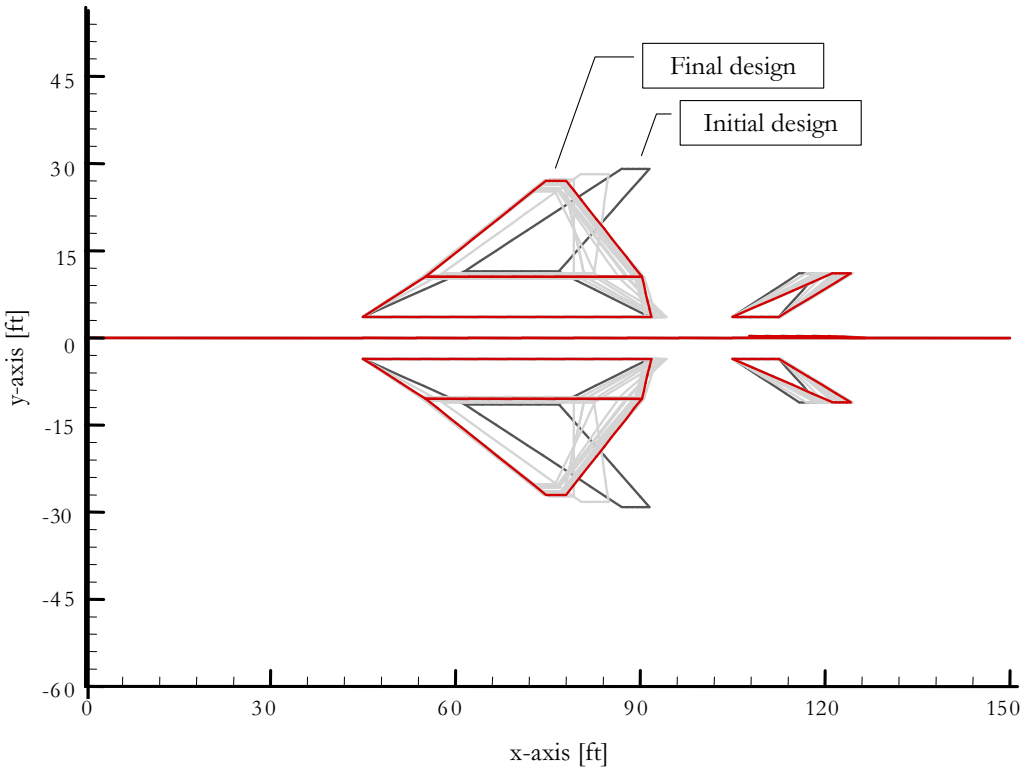

Figure 73: Optimization of Third DOE Case

Table 18: Optimized Properties for Third Case

Property	Start Value	Optimized Value	Unit
Span	64.0	59.8	ft
Aspect Ratio Entire Wing	2.95	0.97	-
Taper Inner Wing	0.30	0.75	-
Taper Entire Wing	0.10	0.07	-
Sweep Inner Wing	70.0	55.0	deg
Sweep Outer Wing	55.0	50.0	deg
Theoretical Wing Area	1393	3688	ft^2
Thrust Loading	0.75	0.75	-
Thrust	112614	104945	lb
Wing Loading	107.8	37.9	lb/ft^2
Take-Off Gross Weight	150152	139926	lb

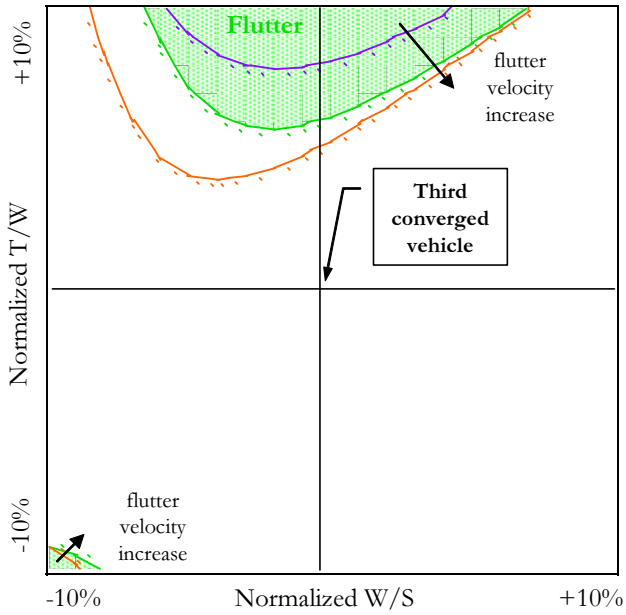

Figure 74: Iso-flutter and Divergence Line for Third Vehicle

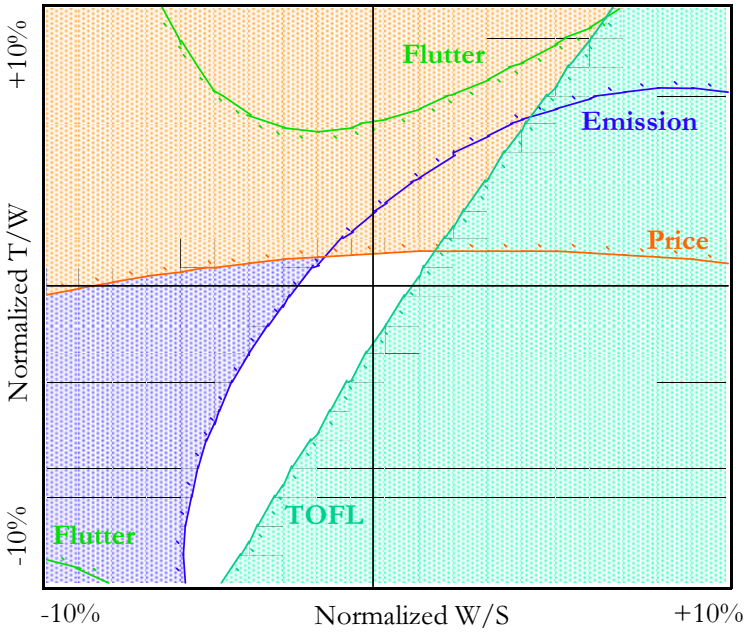

Figure 75: Additional Constraints in the Design Space for Third Vehicle

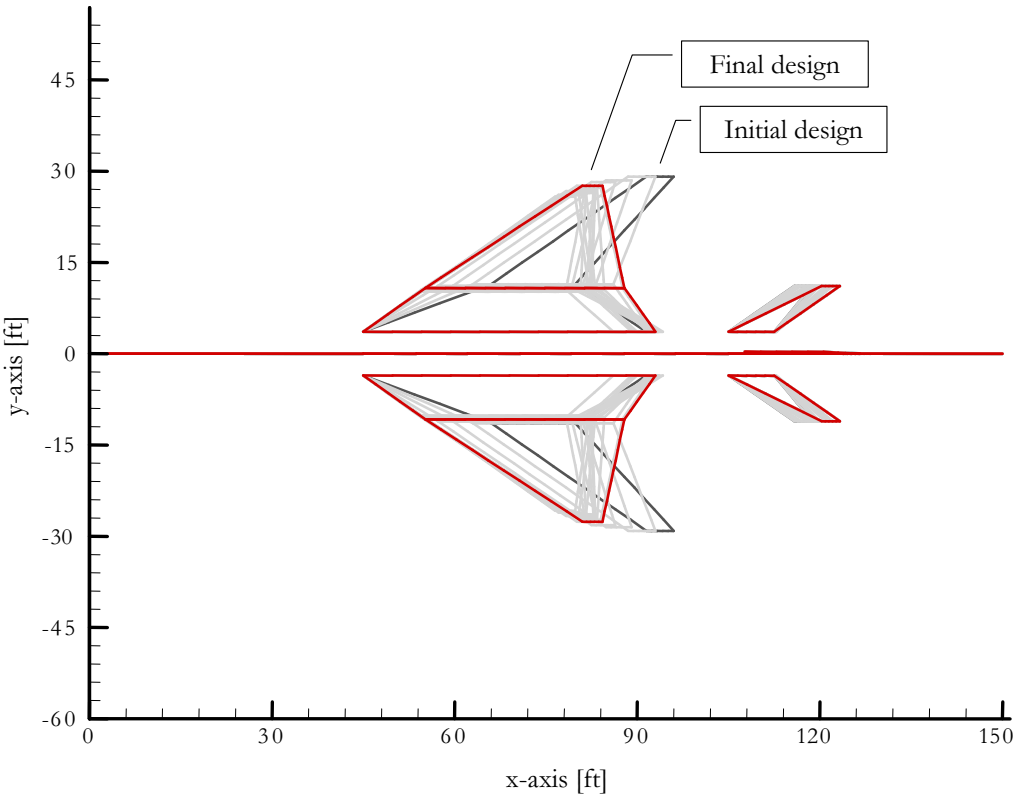

Figure 76: Optimization of Fourth DOE Case

Table 19: Optimized Properties for Fourth Case

Property	Start Value	Optimized Value	Unit
Span	64.0	60.6	ft
Aspect Ratio Entire Wing	2.95	1.07	-
Taper Inner Wing	0.30	0.68	-
Taper Entire Wing	0.10	0.07	-
Sweep Inner Wing	70.0	55	deg
Sweep Outer Wing	55.0	57	deg
Theoretical Wing Area	1393	3421	ft^2
Thrust Loading	0.75	0.75	-
Thrust	117383	106151	lb
Wing Loading	112.4	41.4	lb/ft^2
Take-Off Gross Weight	156510	141535	lb

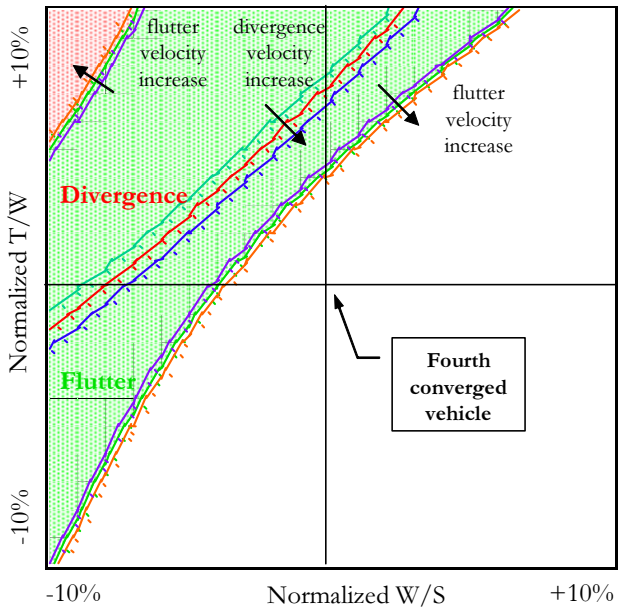

Figure 77: Iso-flutter and Divergence Line for Fourth Vehicle

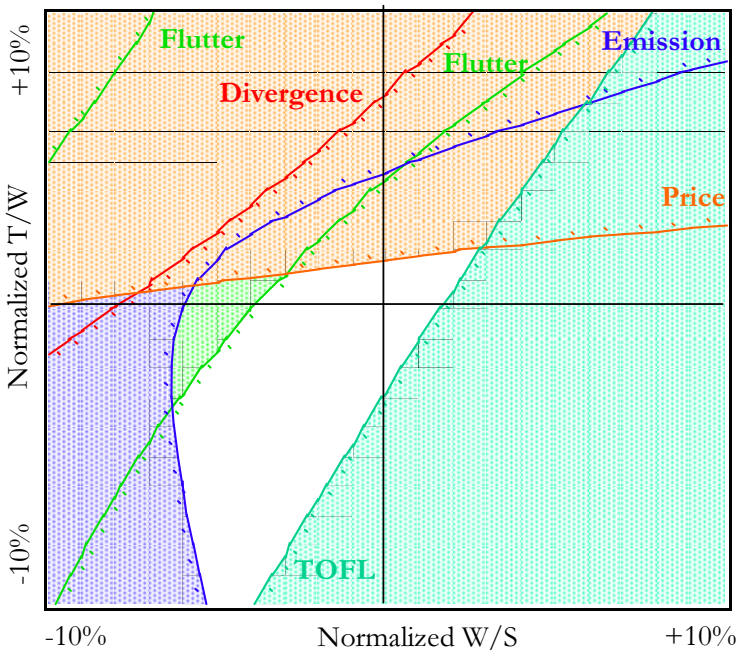

Figure 78: Additional Constraints in the Design Space for Fourth Vehicle

7.5.5 Mission Uncertainty

One of the first observations is the similarity in planforms for all four cases, as shown in Figure 79. In order to try to find a robust design point, the actual flutter and divergence models are much more useful.

The flutter speed models obtained with the scaling approach are summarized in Figure 80. For all four cases these are nearly identical. When thrust loading and wing loading were both high a high flutter speed resulted in all cases. In general high wing loading resulted in a higher flutter speed. This was due to the wing being smaller and thus inherently stiffer and less prone to flutter. The only difference between flutter models was the amount of curvature of the RS.

The divergence speed models showed more significant changes. Especially going from the first to second case, as depicted in Figure 81, resulted in the model being flipped. The fourth divergence model deserves some attention as it differs the most from the saddle points of the other three. When examining the curvature of the slope, it is postulated that the behavior of divergence speed is essentially the same as the first case except that the saddle point was not reached. This contention will be proven when examining the structural weight. In all cases except the first, the divergence speed is highest for the stiffest (smallest) wing, i.e. the high wing loading side.

The effect of varying thrust loading is an order of magnitude less then varying wing loading on both flutter and divergence. This is due to the engines being close to the center of gravity and not having a significant impact on the structural sizing of the vehicle. This in turn means that the eigenfrequencies of the structure are not significantly changed by the engines. And this results in the aeroelastic calculations being less correlated with thrust loading. The minor effect shows that with increasing thrust, the weight goes up. This trend is explained by realizing that increased engine weight and forces ought to result in a beefier structure. For all four cases, the effect of thrust loading and wing loading on structural weight is plotted in Figure 82.

Table 20: Predicted Model Coefficients

Case	1	2	3	4
Mach Number	2.0	2.0	1.5	1.5
Angle of Attack	+2.0	-2.0	+2.0	-2.0
Altitude	10000	60000	60000	10000
Flutter Speed				
b_0	4840505	760264	-6972958	-199565
b_1	-58692	146079	52345	35843
b_2	-9107860	-9883458	15855488	-1516744
b_{11}	1490	696	786	322
b_{12}	-66024	-265816	-171457	-82960
b_{22}	7079173	13632137	-5420033	3323959
Divergence Speed				
b_0	1444276	-597854	1048052	-13848
b_1	25107	16597	30702	-1350
b_2	-5183723	692891	-4964636	255045
b_{11}	-3387	1305	1069	24
b_{12}	326278	-166188	-175460	620
b_{22}	-5117185	4149544	9104364	321999

The thrust loading has especially low effect on the structural weight in cases two and three. The wing loading in cases two and three also has different curvature from cases one and four. From this figure the impression is that altitude is the prime contributor to this effect. However, when investigating the coefficients that make up these RS equations, and making these a linear function of the Mach number, angle of attack, and altitude, an interesting observation emerges: the Mach number and angle of attack are actually what control the coefficients of the RS equations as shown in Figure 83.

The equation coefficients governing the divergence and flutter surfaces, in Figures 80 and 81, may also be written as a linear function of Mach number, angle of attack, and altitude as in Equation 56. These coefficients for the four DOE cases are summarized in Table 20.

$$\mathbf{b} = f(M, \alpha, z) \tag{56}$$

$$V = b_0 + b_1(W/S) + b_2(T/W) + b_{11}(W/S)^2 + b_{12}(W/S)(T/W) + b_{22}(T/W)^2 \quad (57)$$

Graphs can now be made by introducing these predicted coefficients from Equation 56 in Equation 57, which is the quadratic RS equation for speed; where b_0 is the intercept, b_1 refers to the wing loading, and b_2 to the thrust loading. By entering the same parameter conditions as for the four DOE cases, Figures 84 and 85 can be made. The models obtained this way predict the curvatures and slopes accurately.

Figures 86 and 87 illustrate the differences between actual and predicted models in more detail. The prediction of flutter speeds is markedly better then for divergence speed. The mean of the flutter model speeds is much closer to the 45 degree line than of the divergence speeds, as shown in Figure 86. The residual of the flutter speed is showing a slight over-prediction. The mean for the divergence speed is off by about the same amount as the flutter speed but is under-predicting. The difference between the actual and predicted speeds is called the residual. The spread of speeds and associated residuals is shown in Figure 87.

As with the structural weight equation coefficients, the coefficients for the flutter and divergence speed are also simply functions of Mach number and angle of attack. Altitude only plays a minor effect on these coefficients as illustrated in Figures 88 and 89.

Lastly, the design space may be examined. To see the distribution of vehicles as a function of the three parameters, a MCS (Monte Carlo Simulation) approach is used. Uniform distributions for all three parameters can be assumed as shown in Figure 90. This means that any number between the ranges specified has the same probability p of being chosen when these distributions are randomly queried. The wing loading was also made a function of the Mach number, angle of attack, and altitude. The thrust loading also had a uniform distribution associated with it.

Using this MCS approach, 10000 randomly generated cases (randomly picking of

Table 21: Percent of Vehicles by Altitude Range

Altitude Range	Percent of Vehicles
10000 to 22499	22.3
22500 to 34999	25.0
35000 to 47499	25.0
47500 to 60000	24.4
Total	96.8

numbers according to the uniform distributions) are run through this process. From the data generated, the points which had a positive flutter and divergence speed were selected. Almost ninety seven percent of all 10000 points simultaneously satisfied these two requirements. These were then sorted according to four altitude ranges between 10000 and 60000 ft. The distribution of points is shown in Figure 92 and Table 21.

These points are well distributed except for the low altitude case. This plot shows that the flutter and divergence speed are independent of angle of attack and (supersonic) Mach number. The design space is constrained at the low altitude (less data points) but quickly opens up with increasing altitude. This is a commonly known property of flutter that the low altitude case tends to be one of the design cases in flight testing. Figure 91 shows that the number of points in the low altitude case is quickly decreasing.

A plot of dynamic pressure q versus Mach number M superimposed with a flutter boundary as was shown in Figure 31 in Chapter 5. This shows that the low-altitude, high-subsonic case is usually the most critical case in the flight envelope.

A very important remark is also needed with respect to the importance of the angle of attack in these results. The angle of attack should not show up as important in an aeroelastic analysis. The fact that it is showing as important in this research, is an artifact form the approximation processes, and indicates that more accurate models and data points are needed.

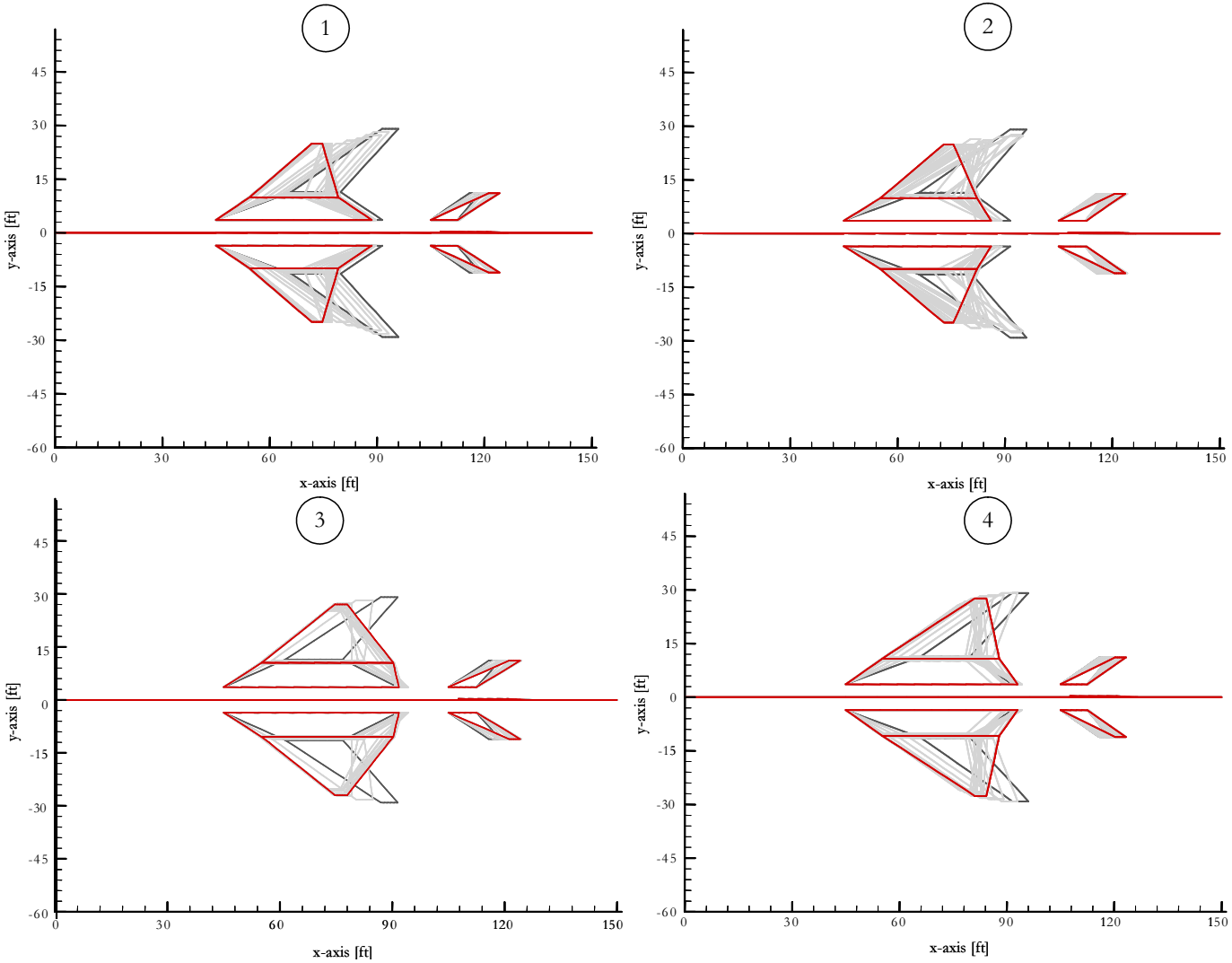

Figure 79: Comparison of Optimized Vehicles

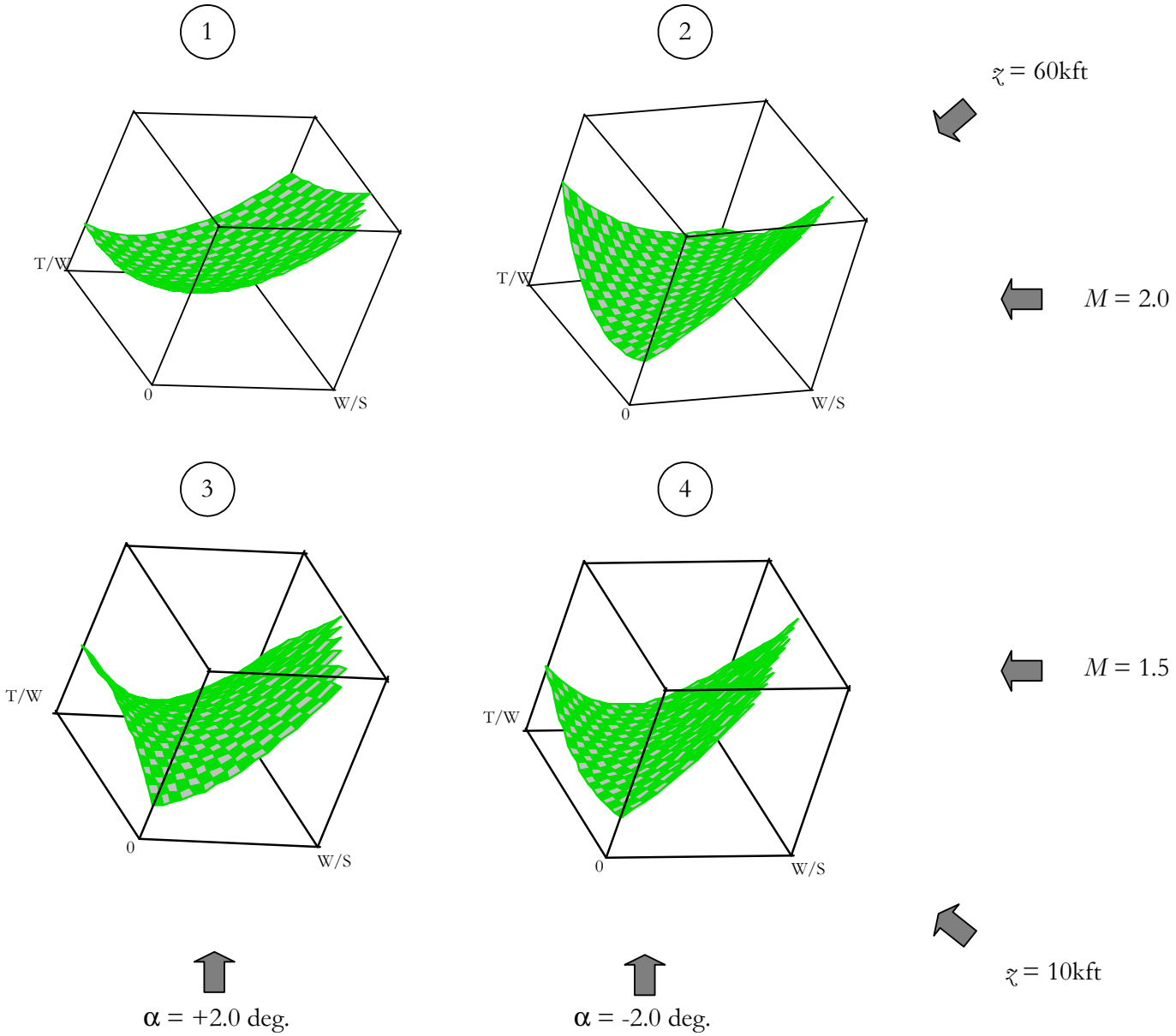

Figure 80: Model Comparison for Flutter Speed

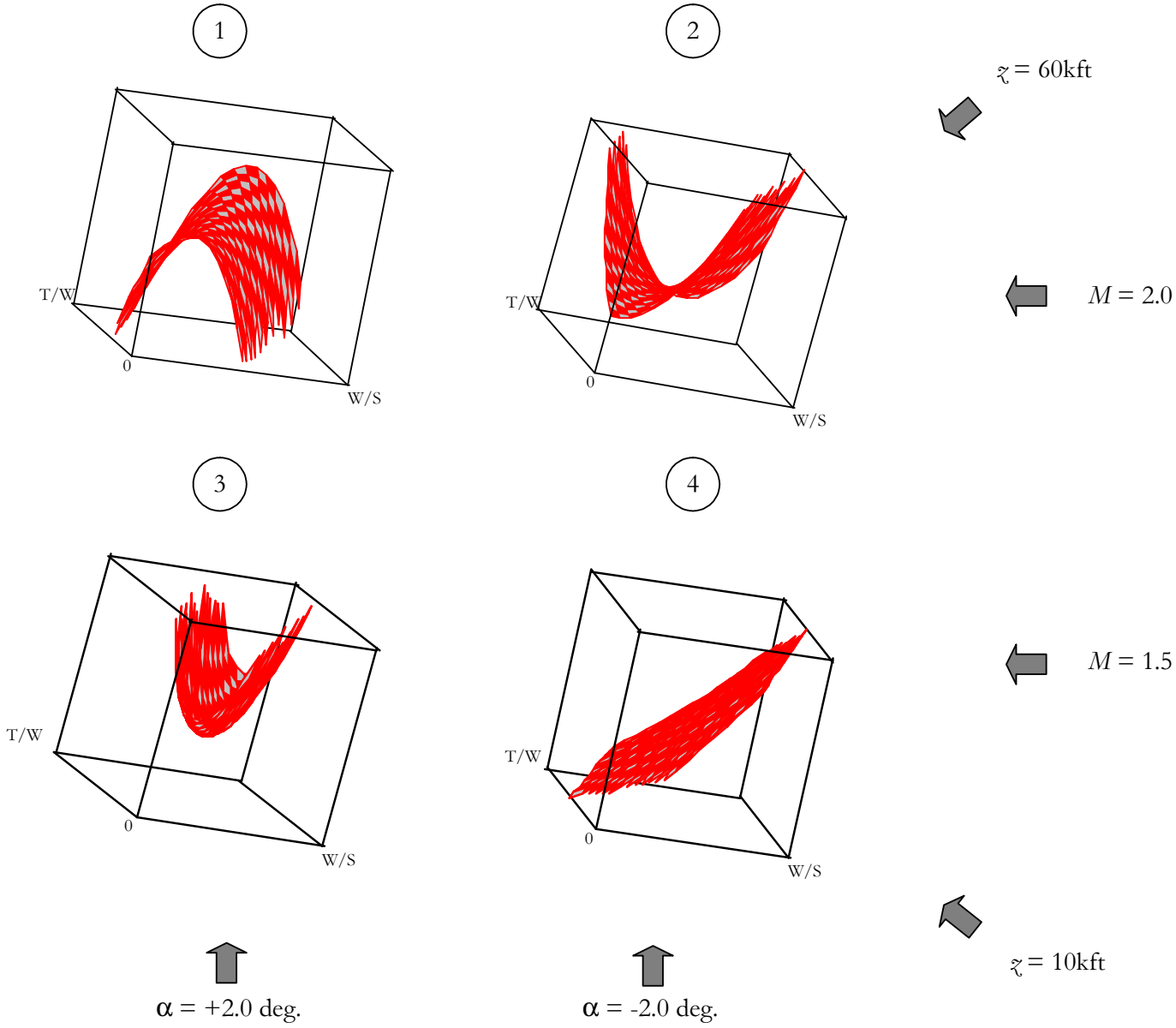

Figure 81: Model Comparison for Divergence Speed

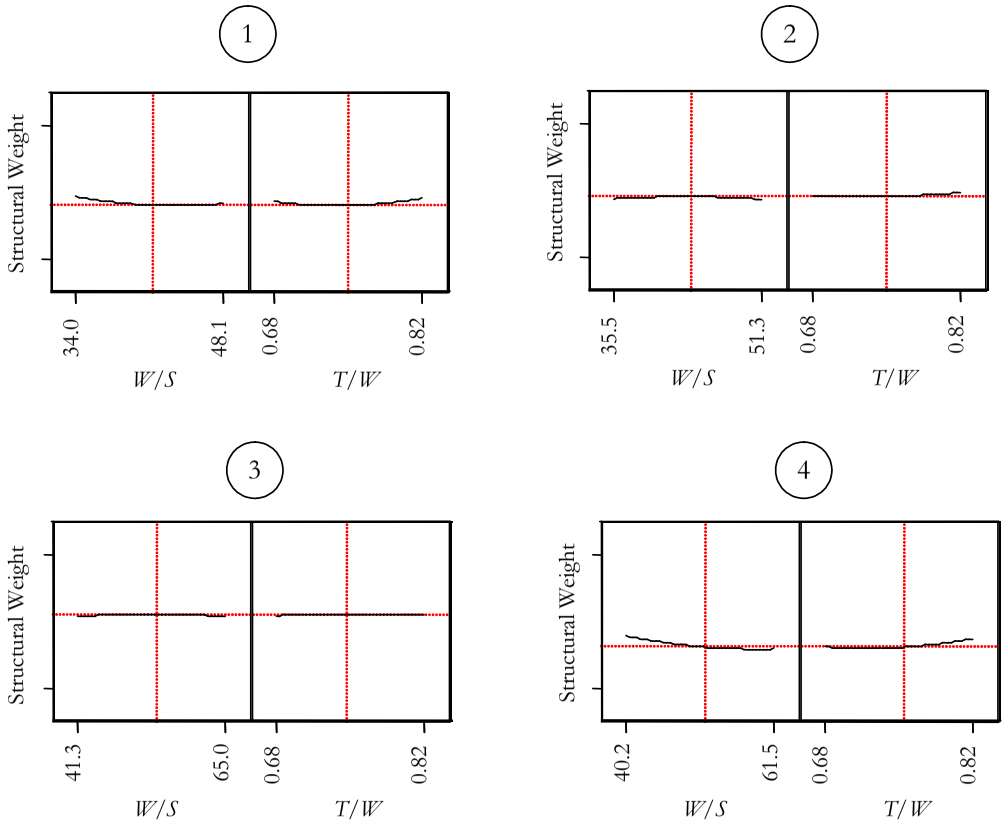

Figure 82: Structural Weight as a function of Wing Loading and Thrust Loading

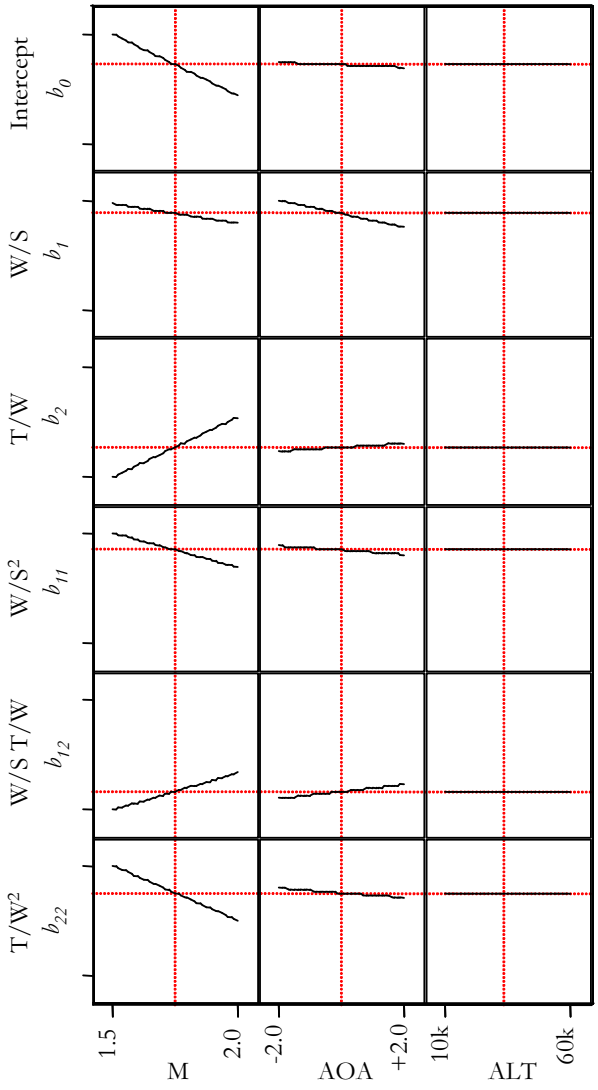

Figure 83: Coefficients of Structural Weight Response Surface as a function of Mission Parameters

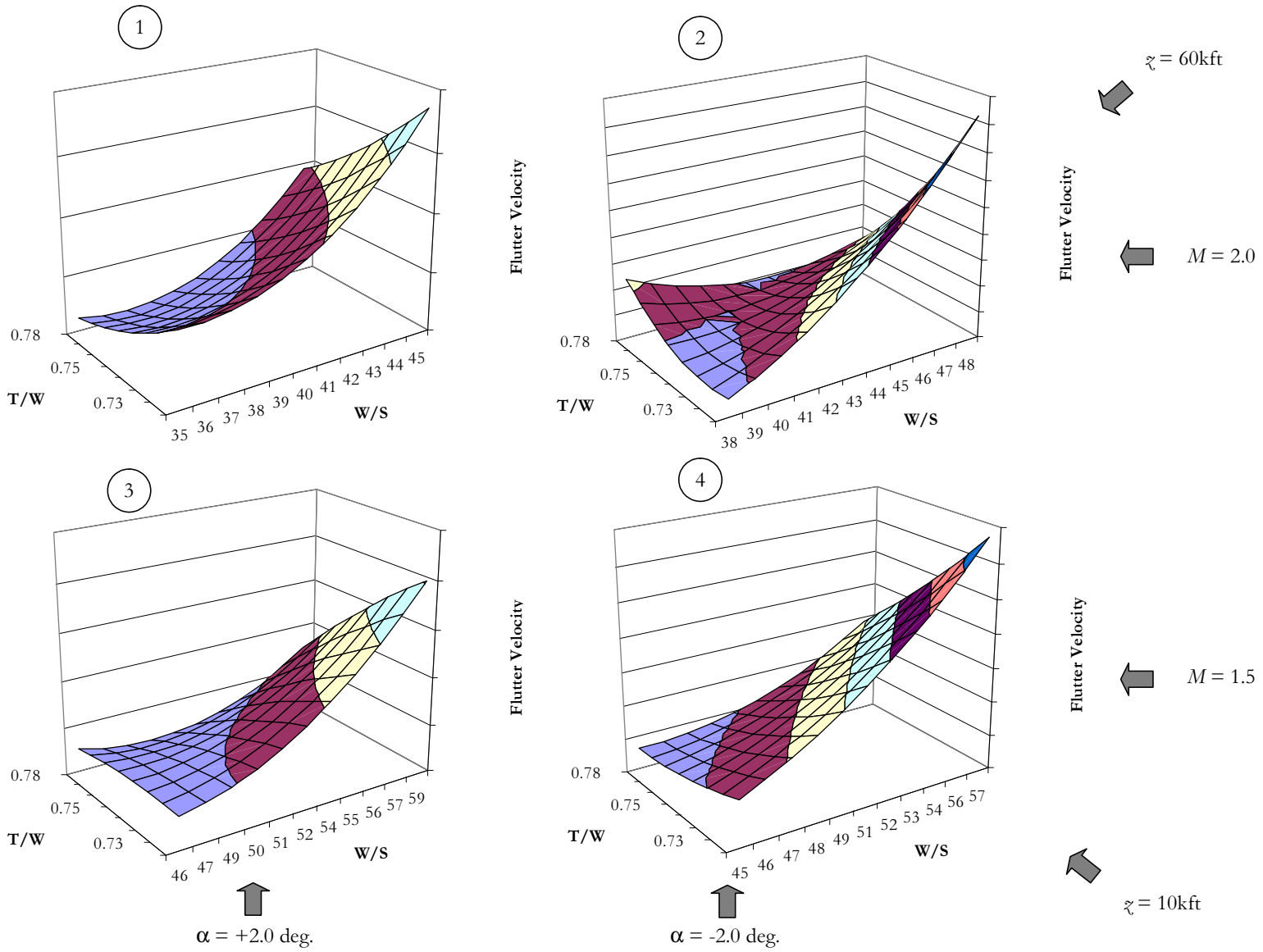

Figure 84: Predicted Models for Flutter Speed

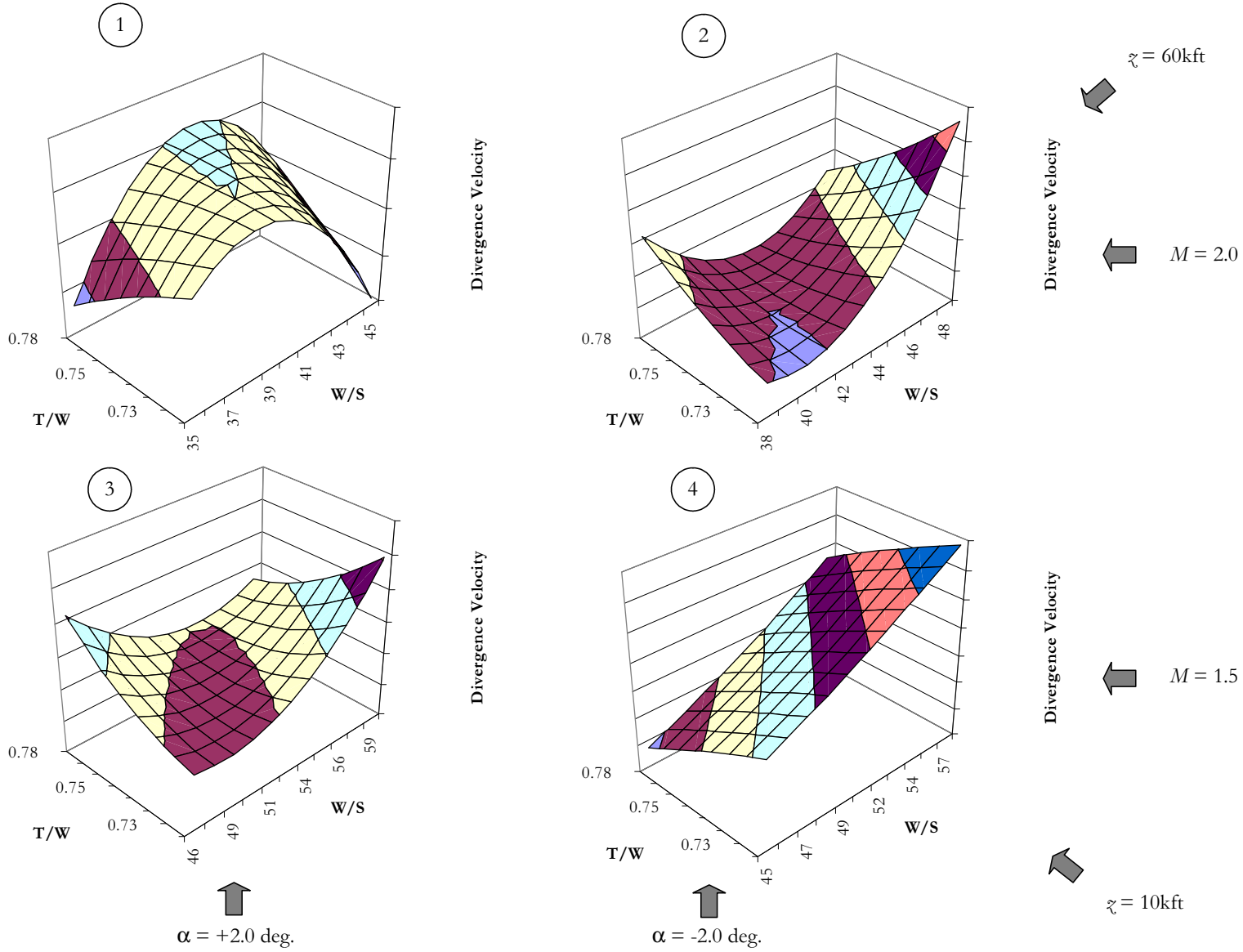

Figure 85: Predicted Models for Divergence Speed

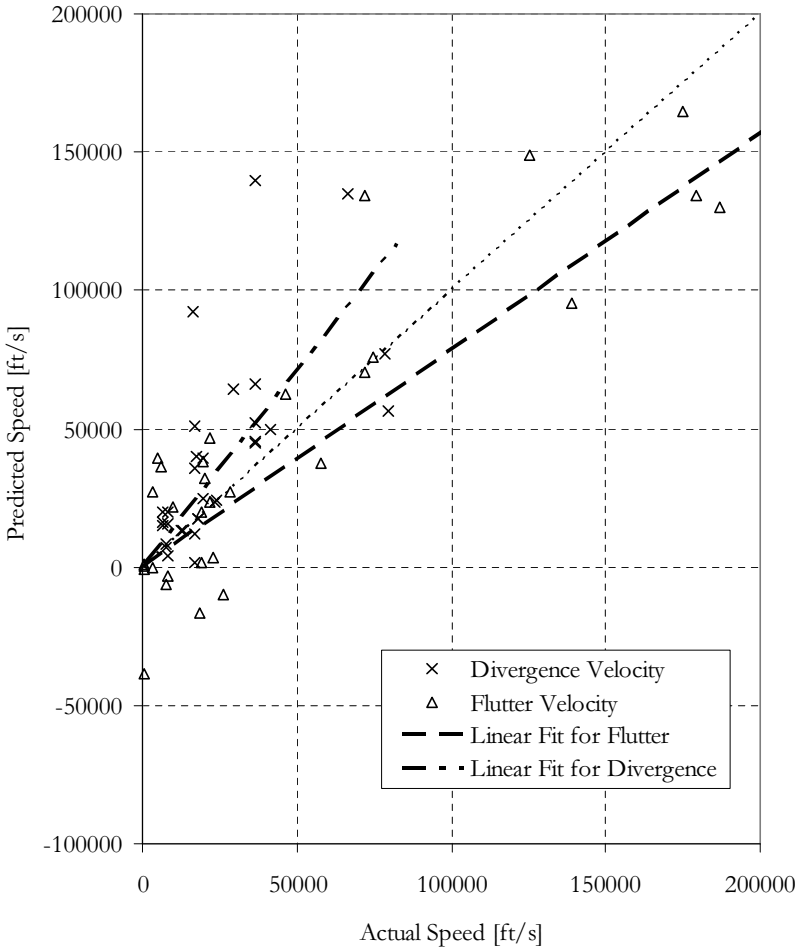

Figure 86: Actual versus Predicted Speeds

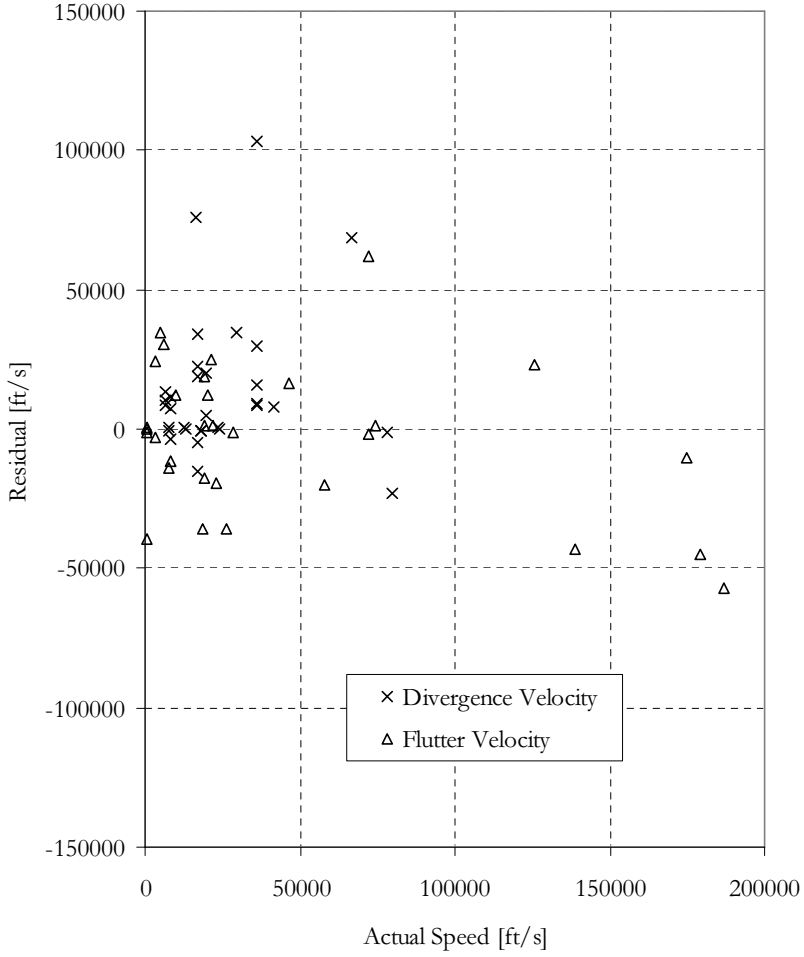

Figure 87: Actual Speeds versus Residuals

Figure 88: Coefficients of Flutter Response Surface as a function of Mission Parameters

Figure 89: Coefficients of Divergence Response Surface as a function of Mission Parameters

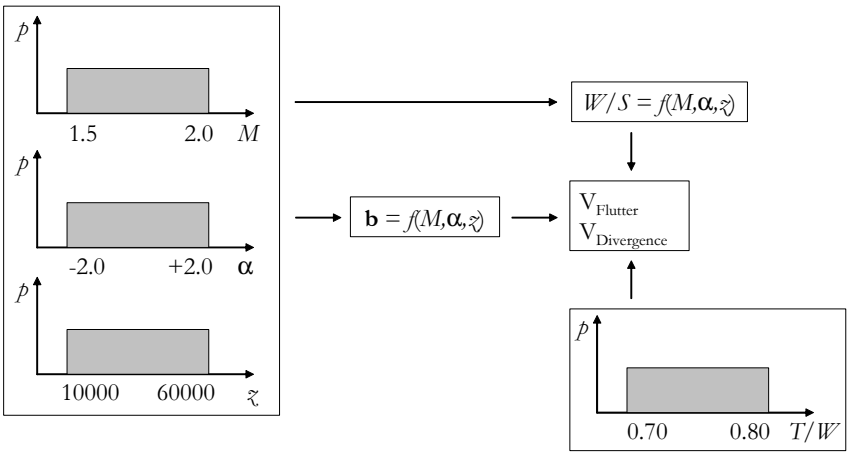

Figure 90: Monte Carlo Simulation Process Flow

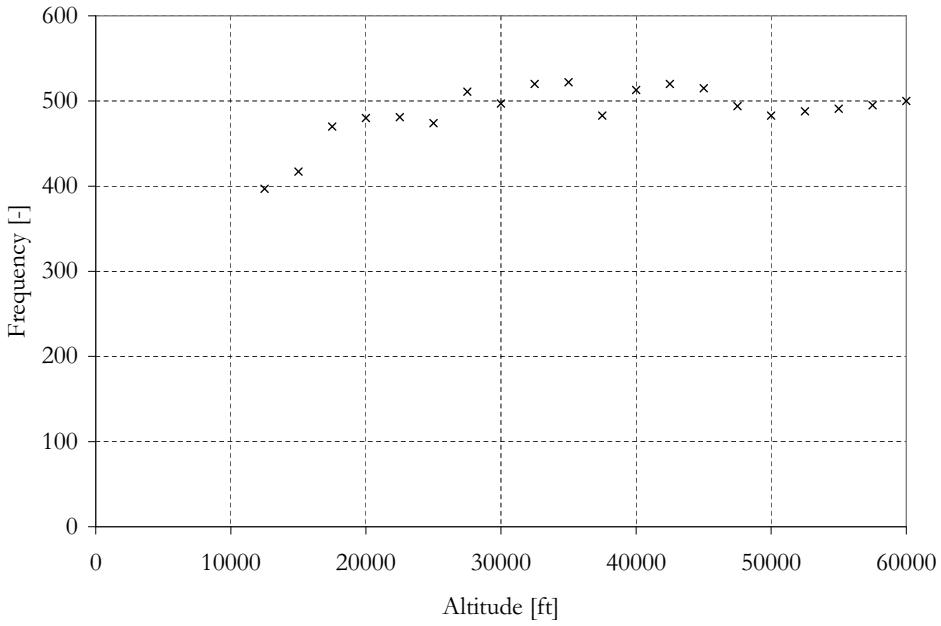

Figure 91: Monte Carlo Points Frequency in Altitude Histogram

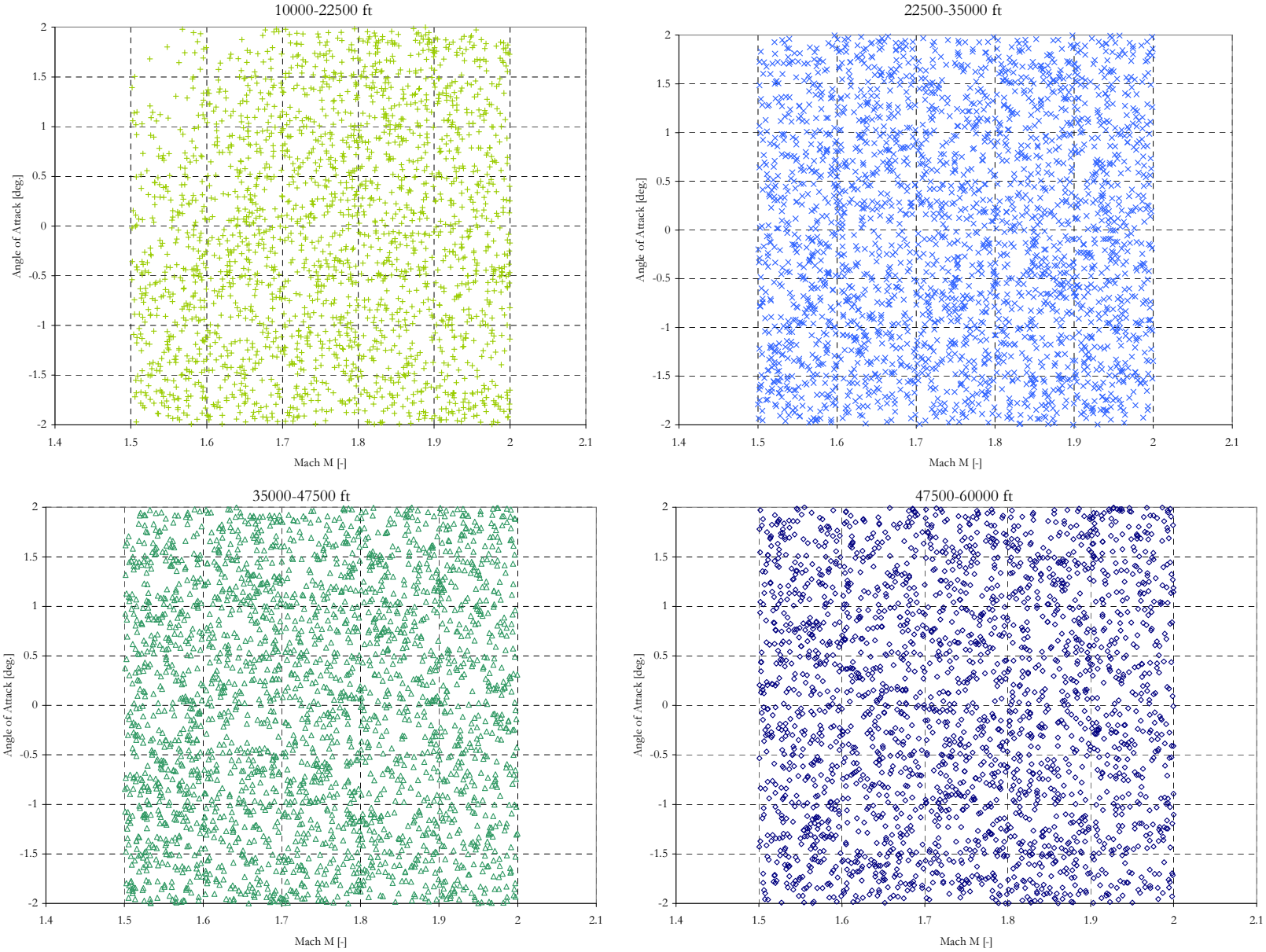

Figure 92: Monte Carlo Points in Design Space

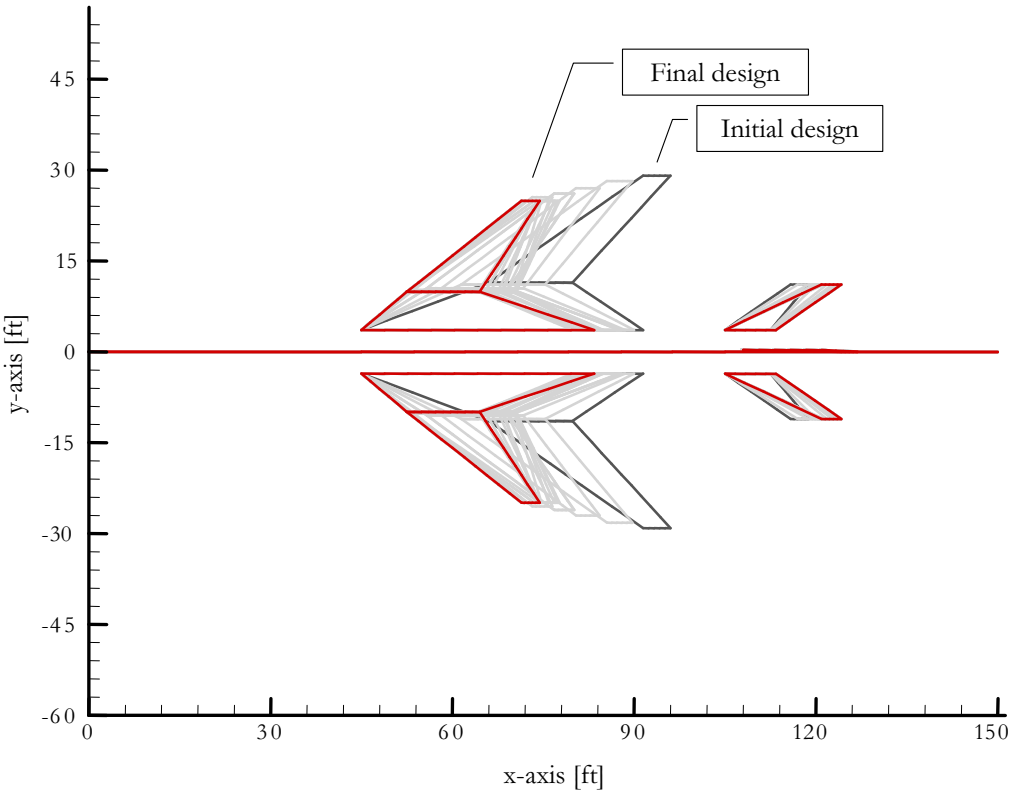

Figure 93: Subsonic Vehicle after BLISS Optimization

7.5.6 Subsonic Comparison

In order to accurately capture the atmospheric uncertainty using only three variables and minimizing the number of runs, a linear main effect DOE was used. Because of the significant changes in physics from subsonic to transonic to supersonic, a main effects model would not be appropriate to cover the entire region. The DOE of main effects was concentrated on the supersonic region. The linear interpolation between the different altitude, speed, and angle of attack in the supersonic region was considered a good zeroth order approximation. Since all the cases were run in the supersonic region, the method's validity was untested at the other speed regions. For this purpose, a subsonic vehicle was optimized with the method, in hope of verifying some common trends if only subsonic aeroelastic constraints were enforced.

Table 22: Optimized Properties for Subsonic Case

Property	Start Value	Optimized Value	Unit
Span	64.0	54.6	ft
Aspect Ratio Entire Wing	2.95	2.73	-
Taper Inner Wing	0.30	0.31	-
Taper Entire Wing	0.10	0.07	-
Sweep Inner Wing	70.0	50.0	deg
Sweep Outer Wing	55.0	52.0	deg
Theoretical Wing Area	1393	1090	ft^2
Thrust Loading	0.75	0.75	-
Thrust	70760	65899	lb
Wing Loading	67.7	80.6	lb/ft^2
Take-Off Gross Weight	94347	87865	lb

The BLISS optimization results can be found in Figures 147 through 155 in Appendix D. A summary of the optimized planform characteristics is listed in Table 22. Figure 93 shows the evolution of the planform from the same starting point as before. An interesting result is the convergence to a high aspect ratio outer wing. Due to the fixed location of the inner to outer wing junction, it is unclear if the optimization would try to move this location. Also due to some fixed geometry variables the optimization was not completely free to alter the geometry. Nevertheless, the different trends from the supersonic cases are visible.

The system optimizes to a high-aspect-ratio wing, as is sufficient for subsonic aeroelasticity. This case removed the aeroelastic supersonic requirement and verified a low speed aeroelastic bound. This changed the delta-shaped wing to a high-aspect ratio wing. Because this wing changes beyond a simple perturbation of the original starting point, the synthesis and sizing code drag polar is invalid. As with the other cases so far, the wing was not penalized for the decreased sweep. This however already shows a different trend towards a higher aspect ratio outboard section. The next section will verify what the impact is of the fixed drag polar and how that changes the design.

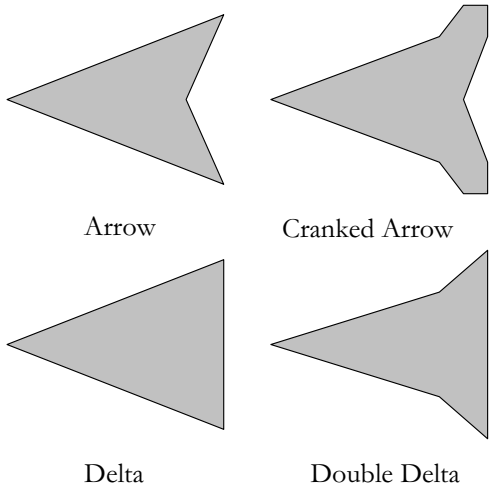

Figure 94: Supersonic Business Jet Planform Variations

This case proves another possibility of the method. Usually a vehicle is designed to a specific mission, which is performed for a fixed cruise speed. The vehicle however needs to be flutter and divergence free in a much wider envelope. In all cases until now, the flutter analysis was done at Mach numbers 1.5 and 2.0. The aeroelastic analysis should however be re-done at different speeds. All the aeroelastic characteristics now obtained, should then be included in the system optimization. The obtained planforms would then be optimized to cruise at a certain speed and flutter/divergence free operation guaranteed in other parts of the flight envelope.

7.5.7 Higher Fidelity Aerodynamics Comparison

The QSBJ vehicle has undergone much scrutiny in recent years. Most planforms studies have focused on the cranked arrow and double delta. These two are in essence perturbation about the same idea of a highly swept inboard section combined with a less swept outboard section for improved low speed performance. Some combinations are depicted in Figure 94.

Making this observation resulted in the belief that taking a converged QSBJ design, which had passed through many different conceptual design cycles, and using

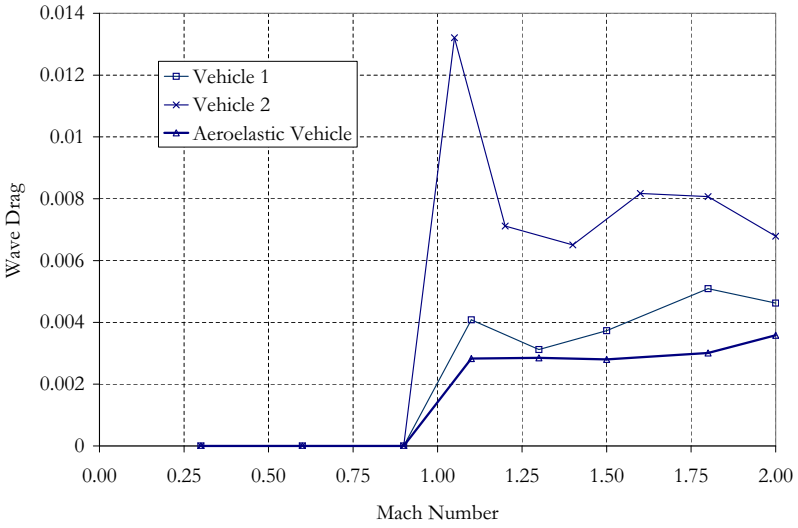

Figure 95: Wave Drag Comparison of Baseline and Aeroelastic Vehicles

different starting points, would not deviate much from this baseline.

With the previous observation in mind, the decision was made that the pre-generated drag polars should not have to be updated. Due to the complexity of implementing BLISS for the first time, it was decided to not add this aerodynamic preprocessing step.

Many times now, the attention was drawn to the fixed drag polar in the synthesis and sizing code not accounting correctly for their radically different shape. During the study, on-going research by Buonanno [19] made available a packaged MATLAB program. This package was able to generate low-fidelity models of VORLAX (Generalized Vortex Lattice Code) [76] and AWAVE [70] to estimate drag polars of a specific vehicle and incorporate updated drag polars in FLOPS-ALCCA. This package was included in the FLOPS-ALCCA setup as a verification of the assumption impact.

The BLISS optimization was repeated for the first DOE case with a Mach number of 2.0, angle of attack of +2.0 degrees, and altitude of 10000ft. This resulted in a planform as shown in Figure 96 and a summary of the optimized planform characteristics is listed in Table 23.

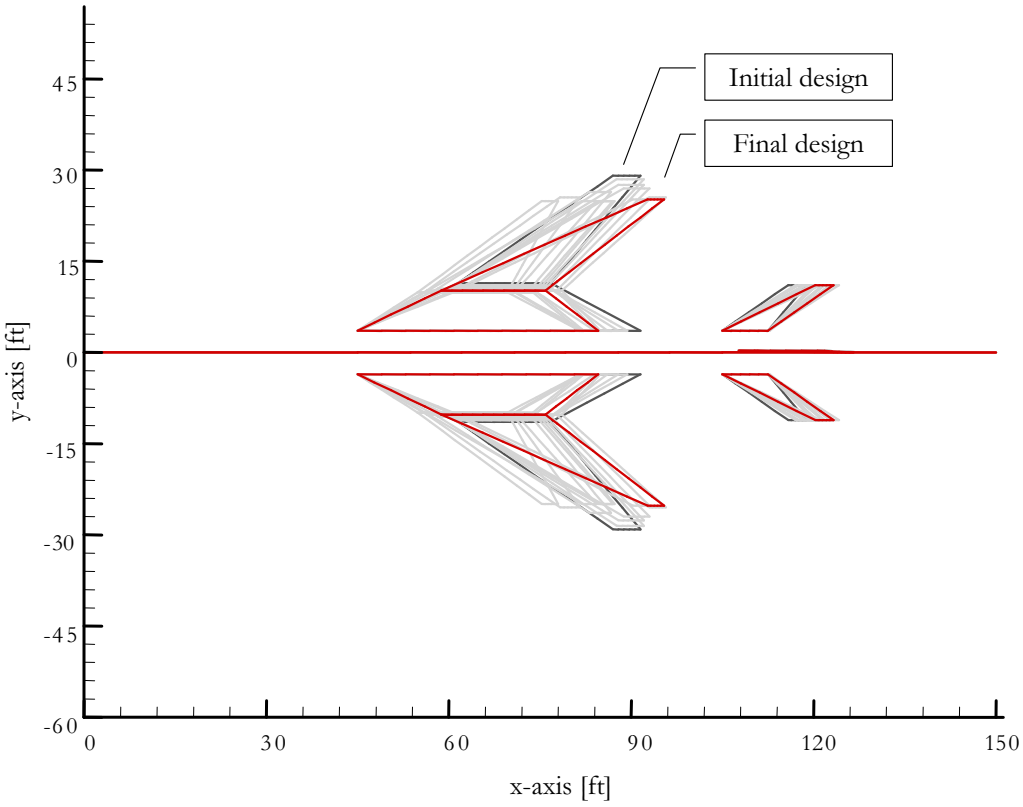

Figure 96: Optimization of High-Fidelity Aerodynamics Case

The highly swept inboard and outboard sections were preserved. The aeroelastic vehicle has an optimized weight of 111776 lb, which is very close to the two baseline vehicles mentioned in the opening sections of this chapter. These had an optimized take-off gross weight of 125381 and 120659 lb. It seems from this that the penalty to include aeroelastic satisfactory behavior is not detrimental to the weight.

The optimized vehicle is characterized by a much lower thrust loading of 0.55. The wave drag of the aeroelastic vehicle is compared to the other two in Figure 95 and shows close resemblance to the first vehicle without the blunted nose.

Table 23: Optimized Properties for High-Fidelity Aerodynamics Case

Property	Start Value	Optimized Value	Unit
Span	64.0	55.3	ft
Aspect Ratio Entire Wing	2.95	1.88	-
Taper Inner Wing	0.30	0.43	-
Taper Entire Wing	0.10	0.07	-
Sweep Inner Wing	70.0	65	deg
Sweep Outer Wing	55.0	66	deg
Theoretical Wing Area	1393	1625	ft^2
Thrust Loading	0.55	0.55	-
Thrust	79680	61477	lb
Wing Loading	104.0	68.8	lb/ft^2
Take-Off Gross Weight	144872	111776	lb

7.6 Discussion of Results

It was observed that all four supersonic vehicles resulted in delta wings. These results were all obtained with a fixed drag polar. The inclusion of higher fidelity aerodynamics, which updated the drag polars, preserved the high sweep configuration.

Of interest is other research by MacMillin et al. [64], which included additional practical constraints and obtained planforms very similar to the delta shapes. That research used the same high-fidelity aerodynamics as was used in this research's higher fidelity aerodynamic case. So, in both studies, there are drivers that are pushing the wing trailing edge forward, away from the initial starting point. Due to the limited inclusion of practical constraints in this research, this shape is lost in the higher fidelity aerodynamic case.

The initial design and starting point of this research is essentially a design that originated from aerodynamicists over forty years ago. As was suggested in the opening chapters, this ignored structural and aeroelastic effects. When imposing structural effects on this aerodynamic, initial design, the wing was swept forward. This move forward was needed to move the wing center of pressure closer to the maximum

wing thickness. This addressed the otherwise significant wing bending moment more effectively. The price paid of this less swept wing is greater drag.

The inclusion of horizontal and vertical tail sizing, engine location limits, undercarriage loads, and control and stability however, should only amplify the importance of the less swept wing. Another argument in favor of this hypothesis is that the newest fighters, F-22 and Joint Strike Fighter, all ended up with diamond planforms very similar to the delta shapes.

Another similarity with these diamond-shaped planforms is the sharp taper. This reduces the bending loads across the wing because most of the lift is generated close to the root. The system optimizer in this study also captures this trend.

The composite objective function weight used in the structural optimization objective function resulted in weight and deformations mostly being weighted near equally. The change in bias oscillated between 0.4, penalizing deformations more, and 0.6, penalizing weight more.

Another observation is the way in which aeroelasticity limits the design space. The accuracy of the models does not allow a statement to be made with a high degree of certainty and further investigation is desired. The under and over-prediction errors of the model are also a sign that more points are needed in the top-level BLISS DOE. The models still seemed to be capable of predicting that the lower altitudes are the most constraining, as is predicted from theoretical investigations [13].

A note can be made concerning the structural weight equations in FLOPS-ALCCA which are essentially based on regressed data. FLOPS-ALCCA is predicting the weights based on the load paths. Unlike the changes in physics of the aerodynamics when going from subsonic to the supersonic regime, the structural physics are not significantly altered; the load paths remain the same.

Supersonic and transonic vehicles must carry extra loads resulting from higher dynamic pressures and shock waves. It seems that the current structural equations do

account for this effect. More importantly, the equations apparently take into account the aeroelastic effects and weight penalty associated with those. The requirement for stiffer structures is captured in the FLOPS-ALCCA equations for this planform.

This study assumed that the achieved cranked arrow design point, iterated many times by different research in the past years, would be a good initial estimate but perhaps not accounted correctly for aeroelastic penalties. This was proven wrong as there was no significant weight penalty observed with the higher fidelity aerodynamic case. This shows that the regressed weight equations of the current FLOPS-ALCCA implementation are implicitly accounting for aeroelastic constraints for this planform. Significantly different planforms would most likely not obey to this rule and that is where this method would be most useful.

Lastly, the method required a minimal but not trivial form of human input. Any engineering method, especially for conceptual design, will require a person in the loop. It is the hope of such new methods to reduce the human interaction required to a minimal amount, or at least free the user from trivial tasks such that he can concentrate on the more important task of design guidance.

The human oversight was needed for two tasks. The decision on bounding ranges for the RS equations was the first task. These ranges required careful choice since internal code constraints resulted in specific regions of the design space being viable.

The setting of a convergence criterion and at what time a switching of code fidelity was needed, was another task of the designer. The conceptual design phase has specific goals set, and once these are met, higher fidelity tools have to be introduced to increase accuracy and investigate other promising tracks. Human interaction is also needed to decide when a vehicle was optimized to an acceptable level, sufficient for conceptual design purposes.

CHAPTER VIII

CONCLUSIONS AND RECOMMENDATIONS

This research implemented the novel decomposition technique BLISS in an environment characterized by tight couplings and significant data flows. The decomposition approach was found to address effectively the large volume of data that couple structures and aerodynamics. The separation of disciplines and representation of their results in surrogate models used by a system optimizer required a careful choice of data condensation to reduce computational expense. The isolation of disciplines allowed the use of massively concurrent processing power.

8.1 Research Contributions

This research's novelty lies in four points. First is the use of physics-based tools at the conceptual design phase to calculate the aeroelastic properties. This allows for the generation of aeroelastic constraints that are not simply perturbations around a fixed design. This method has the flexibility to allow significant changes in the concept's geometry.

Second is the projection of flutter and speed constraint lines in a thrust loading versus wing loading graph. This is a unique result for the design community. The mapping of such constraints in a designer's familiar format is a valuable tool for fast examination of the design space.

Third is the aeroelastic assessment time reduction. Until recently, because of extensive computational and time requirements, aeroelasticity was only assessed at the preliminary design phase. This research illustrated a procedure whereby, for the first time, aeroelasticity can be assessed at the conceptual design formulation stages and

results obtained in a timely manner. Further increases in massively concurrent data processing power can be immediately exploited with BLISS. The BLISS decomposition technique can intrinsically engage multiple processors concurrently.

Forth, this assessment was robust as the impact of changing speed, altitude and angle of attack could be immediately verified. In such a way, critical areas in the design space could be identified.

8.2 Research Hypotheses

The hypotheses posted in the opening chapters of this document are recalled and their correctness verified.

Hypothesis 1 – Decomposition techniques in conjunction with physics-based analysis codes can be used on a large-scale, time-expensive problem.
Hypothesis 2 – BLISS can decompose and manage these data flows between physics-based analysis codes efficiently.

These two hypotheses were correct. The BLISS decomposition technique did allow the use of physics-based codes at the conceptual level. The data flows are manageable although careful consideration is needed of where aeroelastic constraints are positioned. Improvements by including other disciplines and more detailed models would not limit the applicability of BLISS. Computer resources were modest but it was proven that the method with more computational power would allow the incorporation of additional disciplines.

Hypothesis 3 – RS in conjunction with DOE can accurately capture the design space trends.
Hypothesis 4 – The initial wide ranges and resulting inaccurate RS equations do allow for convergence of the optimization to occur.

These research hypotheses were partly true, partly false. The DOE and RS method allowed convergence of the system optimizer, but questions were posed as to the accuracy of this quadratic model. One of the reasons was the potential existence of flutter hump modes. Other solutions may be found in investigating different starting points, or different surrogate models such as ANN or kriging. Both solutions are only feasible with an increase in computer resources. The ANN approach is the most computational expensive alternative and would likely only be practical with use of even less detailed conceptual models. Research into surrogate modeling and approximation techniques could be immediately applied in BLISS as it is an ideal receptacle for that.

Hypothesis 5 – By scaling of the wing and thrust of this one BLISS optimized vehicle point design, an aeroelastic constraint line may be formed.

This scaling hypothesis proved correct. The research identified a way to determine how the aeroelastic properties behaved in the neighborhood of an optimized point design vehicle. The accuracy of these lines solely depended on the ranges chosen to scale the vehicle.

Hypothesis 6 – Executing BLISS for different points in the mission allows to see the main effects of these mission characteristics in the thrust loading versus wing loading chart. In such a way, a point design may be chosen that is robust to these variations.

Due to the chosen linear DOE, only main effects were investigated and the region of application was limited to the supersonic regime. This did not prevent the research from answering positively to this hypothesis: it was possible to identify regions of the design space that were robust to a change in these mission parameters. A more accurate DOE is needed to create better (quadratic) models, nevertheless the method

proved to allow the selection of a robust design point while satisfying aeroelastic constraints.

8.3 Recommendations

In this first implementation the assumption was made to not consider loads on the structure and solutions were discussed that show that these could be added with few modifications. To include these loads, control surfaces, and a trim analysis would entail a significant added contribution and require more computer resources. These assumptions worked well to reduce the initial scope of the problem and in helping to allow the implementation and identification of regions were more work is needed. One part of the future work should thus be focused on removing the assumptions that are still present in this work.

More detailed models including the control surfaces would be a first improvement. This would lead to a trim analysis to be performed and due to this a g-maneuver could be specified instead of angle of attack. Another advantage would be more correct load assessment and these could then be transferred to the structural model. A detriment of this assessment is the increase in potential design variables. Some design variables were fixed during this study's optimization, and in order to decrease or at least keep this number of fixed variables constant, more computer resources are needed. The inclusion of the control surfaces and more computers are one of the steps to better results.

The additional disciplines would then help in tracking more practical constraints and answer the remaining question why the delta planform was not preserved when including higher fidelity aerodynamics. The example of the latest fighters and other research, characterized by the move from a cranked arrow to a diamond shape, could then be confirmed.

A second part of the future work should focus on some early choices, which proved

not to be the most ideal. Two choices should be re-thought in future work.

- Due to the impractical corner point cases of the DOE and the choice of Mach speed as a free mission parameter, the design space was severely skewed when plotting the results as a dynamic pressure. In a next iteration the flutter and divergence speed constraints should also have units of dynamic pressure.

- The drawbacks of the free mesh in the structural model were that the node locations were not fixed. Every time a new structural geometry was generated, a different mesh resulted with a different amount of DOF. In order to improve the coupling between the aerodynamic and structural RS equations, the eigenmodes ought to be approximated. The ability to do this requires that the node locations are fixed and thus a mapped mesh be used.

Other future work could concentrate on a plethora of ideas. By having this capability to effectively generate aeroelastic data at the conceptual level, many until now impossible games can be played.

- By using an integrated framework and the flexibility of BLISS, the introduction of additional constraints is feasible. These include noise codes such as PBOOM [21], higher fidelity aerodynamic codes, stability derivatives by using HASC95 [6], etc. Many of these metrics should also be included in the subsystem objective functions such that the aircraft is satisfying as many realistic constraints as possible.

- The performance of other surrogate models can be investigated: linear versus quadratic models, smaller versus wider validity ranges, kriging and ANN surrogate models. The trade-offs to be made in all of these are accuracy versus convergence speed. The less accurate models need smaller operating ranges but more model iterations are perhaps required, increasing the overall time spent generating models.

- Kriging models would result in an overall better fit of the multi-modal design space at the expense of more executions. In addition to the different surrogate model options, an investigation into different starting points should be performed. In dealing with a multi-modal design space the accuracy of the quadratic models could be confirmed.

- A modal reduction technique as was proposed by Karpel [48] could be implemented within ANSYS. This would allow for more detailed models to be used, while maintaining the execution times of the structural dynamic calculation at a reasonable level.

- Due to the sensitivity of the synthesis and sizing code to certain variables, sometimes finding the limits of this code were not trivial. If the process is to be automated more, or the designer's task made easier, then a design space bounds exploration technique is needed. It is unclear where these bounds lie in the multi-dimensional space examined here.

- The research assumed no flutter hump mode and an envelope RS fitting technique was used. That took care of the slope discontinuity in the flutter constraint and deliberately ignored the value discontinuity. In future iterations, the possibility of hump modes should be looked at in more detail by tracking multiple flutter mode constraints.

- Variable complexity or variable fidelity modeling research would be modular to BLISS [38]. The use of surrogate models already allows that the system optimizer does not see the underlying codes. This field of study then has the capability of enhancing the BLISS optimization results effortlessly.

- Lastly, an assessment can be made of the extra weight required for flutter and

divergence free operation while also satisfying performance metrics. Other questions, including penalties of strengthening an already optimized vehicle to obtain aeroelastic satisfactory results versus a entire re-design of the planform with aeroelasticity in mind from the outset, could be answered.

Lastly, there are some recommendations about using BLISS in an industrial-sized process and some other encountered difficulties.

- The number of global design variables Z must be small due to the use of DOE and the time expense associated with physics-based codes. Concurrent processing power is needed here to achieve all the benefits of the method.

- In order to account for wind tunnel data in the design process, the aerodynamic code currently used ought to be swapped with a data look-up table. For example, an SQL database could be used, and easily included in the model.

- If only one lifting surface at a time was allowed to flutter, the aeroelastic properties could be approximated better. The existence of flutter hump modes should also be investigated.

- If only one half of a symmetric model was used, this would have resulted in half the number DOF and less modes needed. This approach would not work for a non-symmetric vehicle.

- The system optimizer choose a certain track and in this multi-dimensional space there is loss of transparency. It is unclear why certain geometric and local variables were more important than others. There is a need to prove that the design is not stuck in a local minimum.

- The present research was characterized by large data flows compared to previous MDO work. However, these data flows would be small when compared to an

industrial process. The method could handle these flows, but potential problems lay in the condensation of this data to field variables and the inclusion of these in the surrogate modeling technique. A careful choice of the approximation technique is needed when additional constraints, more detailed models, and industrial processes are considered.

8.4 Concluding remarks

For the first time a truly multidisciplinary approach was taken to aeroelasticity at the conceptual design phase. The implemented method has the flexibility to include other disciplines such as noise, higher-fidelity aerodynamics, control and stability and map the resulting vehicle to a designer's preferred environment. The speed in which this can be achieved was unique in the aeroelastic conceptual design community.

The implementation can also benefit from research in many different areas such as variable complexity modeling, surrogate modeling techniques, and massively concurrent data processing.

APPENDIX A

FRAMEWORK BACKGROUND

In this appendix, an overview is given of the different alternatives to integrating computational frameworks. Frameworks need to incorporate the latest computational techniques and more importantly a mind-set emphasizing flexibility, modularity, portability and re-usability.

Distributed object computing extends an object-oriented system which allows objects to interact across heterogenous networks and inter-operate as a unified whole. Integrated computing frameworks are discussed, together with data transport techniques such as XML (Extensible Markup Language) and SOAP (Simple Object Access Protocol) to achieve platform, code and meta-model independent integration.

A.1 Framework Characteristics

Today's engineering designers have come to the realization that no longer can successful designs revolve around the analysis and optimization of a single discipline. But rather, successful designs are now viewed as a balance between competing disciplines. Given the above, accounting for and balancing disciplines through the sharing of data between disciplines becomes a monumental task.

MDO consists of the following principal conceptual components [101]:

1. Design Oriented Analysis: System level designing allows the designer to answer the "what-if" questions. Designers want to know the sensitivity of the design with respect to the design variables.

2. Approximation Concepts: Meta-models allow the designer the ability to bypass

the expensive direct coupling of analysis codes to the design space explorer tool. Common meta-modeling techniques such as RS methodology and ANN can be used instead of these complex disciplinary analysis codes.

3. Mathematical Modeling of System: It is axiomatic that an engineering system is usually modeled by multiple disjoint analysis codes and not one monolithic code. Data reduction techniques may need to be applied if large amounts of data are exchanged between codes.

4. Decomposition: Given that codes analyzed on the same level are often tightly coupled, it is usually preferable (if possible) to decouple the individual codes and let the system level take care of the coupling. Here system decomposition techniques and tools such as GSE, CO, CSSO, and BLISS are used.

5. Design Space Search: Exploration of the design space is the search to find the constrained minimum. Various algorithms such as SLP (Sequential Linear Programming) and SQP (Sequential Quadratic Programming) can be applied. Alternatively, where applicable, algorithms employing stochastic processes (GA (Genetic Algorithm) and SA (Simulated Annealing)) are also an option.

6. Optimization Procedures: System optimization is conducted at the system level. The system optimizer knows which codes to execute and in what fashion. This element effectively ties together the different codes in an execution sequence.

7. Human Interface: Manual intervention in the system design process is important and is not an after-thought. In a well designed environment, the implementation should allow for straightforward, designer intervention. This intervention is needed since MDO often relies on human interaction to guide the process.

A.2 History of Integrating Frameworks

Frameworks try to aid engineers in formulating, solving and evaluating complex design problems. Automation of the design process occurs through the framework. To date several commercial applications and research programs have been developed to aid in the dissemination of information between discipline analyses [39] [40] [41] [79] [80]. Nonetheless, these tools do not always afford the designer the flexibility necessary to implement novel information distribution and manipulation techniques.

A.2.1 Hard-Coded Algorithms

In the earliest frameworks, all disciplinary executables were brought together and execution control was given to a fixed algorithm. There are three variations on this in chronological order: monolithic codes, direct integration, and meta-modeling techniques. Most of the monolithic programs were written in FORTRAN and some examples of these efforts are still around, for example: FLOPS-ALCCA and ACSYNT (Aircraft Synthesis) [1]. A general disadvantage of these systems is the relative difficulty to include higher-fidelity tools as they become available. Since these approaches require a total reconfiguration of the script that controls the execution.

Up to recently, there was no real valid alternative to this *hard-coding* of programs. In the last decade, some commercial alternatives arose on the horizon.

A.2.2 Commercial Applications

In the design and analysis of systems, there are currently several design tools available that allow a designer to efficiently explore and design with the overall system in mind. Commercial efforts have produced AML, iSIGHT, and ModelCenter. These allow for a textual or graphical representation of the data flows between analysis tools, automatic output parsing, input file generation, optimization tools, statistical tools, and result visualization.

These environments have certain, inherent drawbacks. Most notable, they are

not open-source thus not allowing the designer to tailor the tool to exactly meet the needs that the analysis may require. These tools allow for the designer to link codes through the GUI (Graphical User Interface) and do not always give direct access to the underlying core of the tool. Consequently, they are not truly conceptual design tools since they do not allow the investigation of concepts. They only allow for perturbations around a user-provided baseline input file.

An interesting comparison can be made by looking at Microsoft Excel and The MathWorks' Matlab. Excel inherently uses a GUI, the spreadsheet, to enter equations and visualize its output. Matlab on the other hand opens up a library of functions, which can be used from a command input window. Over time, Excel added the capability of Visual Basic, which allowed for more powerful operations comparable to Matlab. Nonetheless, Matlab is still a more powerful and flexible tool since it was conceived as an API. The same comparison is true for integrating frameworks: most commercial applications use a GUI to interact with the user. It is this GUI which makes these tools very user-friendly and easy to use, however, a general API built on solid object-oriented programming is potentially much more powerful.

A.2.3 Open-Source Applications

An alternative solution is to develop a general, systems analysis API. Such a general library of methods would allow the designer to programmatically link and execute any number of analyses and manipulate the resulting data in any conceivable manner [54] [92].

Recent advances in business-to-business data transfer as well as server-client application interfacing have made the task for the engineer to develop such an environment significantly easier. More specifically, it is straightforward to create an API that incorporates object-oriented code wrappers written in Java with uniform, standard data transport tools (XML and SOAP) to create an infrastructure which is both platform

independent and flexible to the needs of the designer.

Starting such a task from scratch would entail a significant programming feat for any designer/programmer. Fortunately, readily available tool boxes for optimization [118], statistics [2] [3], simulation, visualization [10], and web-server applications [36] are pre-written, plug-and-playable, and more importantly, open-source freeware.

These framework tools combined with the use of the agents give total flexibility and modularity. This allows the designer to concentrate on the actual design task. More details on the open-source building blocks will follow in later sections.

A.3 Commercial Applications

Scott [97] gave an overview of the commercial endeavors. The following high-level overview of three commercial packages is now provided.

A.3.1 Adaptive Modeling Language

AML (Adaptive Modeling Language) is developed by Technosoft [111]. AML is built on the philosophy of object-oriented software design and uses LISP as its programming language, which is a fairly uncommon language. Variables are created by instances of some previously defined primitives. When defining formulas, AML automatically keeps track of which variables depend on others. AML has easy and polyvalent graphical visualization capabilities (especially for aerodynamic design).

Some disadvantages from a user friendliness perspective are that a good working knowledge is required of object-oriented programming. The use of object-oriented programming is not necessarily detrimental as will be shown when discussing the open-source requirements. Unfortunately, integration with certain tools (Excel spreadsheets etc.) is not functional yet.

A.3.2 iSIGHT

iSIGHT is produced by Engineous Software [30]. iSIGHT is based on the Multidisciplinary Optimization Language, its own language. Pre-made building blocks are accessible through GUIs so to avoid direct interaction with the underlying language. Logic-based control and optimization boxes are readily available from the GUI environment. Options for parsing input and output files are very extensive. The linking between codes occurs implicitly by using the same variable names. Unfortunately, cross-platform integration of different codes and front-end is not straightforward.

A.3.3 ModelCenter

ModelCenter is made by Phoenix Integration [87]. The front-end interface is called ModelCenter, while in the background the Analysis Server services the request coming from the ModelCenter GUI [86]. The ModelCenter "*web-browser*" can be used from any location to add a resource/code which was wrapped and placed on the Analysis Server.

RS generators, Monte Carlo simulation and stochastic optimization toolboxes were recently added to the basic package. One of the remaining drawbacks of ModelCenter is that multiple instances of a code, also known as parallelization, is currently not supported.

Given the ease of use and polished execution that these commercial packages exhibit, there are drawbacks as well. Most notability, the proprietary nature of the source code makes customization difficult. In-house developed methodologies and strategies can be integrated, but that requires clever interaction through a GUI or the user has to use the provided API in a programmatic environment. Access to source code for extreme flexibility is a very important prerequisite in a conceptual design research environment where these new methodologies are developed and investigated.

A.4 Open-Source Applications

The previous section highlighted some of the advantages and disadvantages of commercial packages. An open-source tool should draw on the strengths and weaknesses listed above. Below are a list of items that are worthy for incorporation and investigating in this to-be created tool.

- Use the sound basis of object-oriented programming (from AML).
- Extensive tools for parsing input and output files (from iSIGHT).
- Logic-based control boxes are pre-written and available as functions (methods in Java) and in the API (from iSIGHT).
- Wrapped codes (also called *agents*) are immediately available form a distributed servers (from ModelCenter).
- Methods for multiple instances of agents for parallelization need to be provided (from a shortcoming in ModelCenter).
- Allow for growth potential when incorporating statistical (from ModelCenter), optimization (from iSIGHT), and visualization toolboxes (from AML).

Distributed object computing extends an object-oriented system which allows objects to interact across heterogenous networks and inter-operate as a unified whole. Integrated computing frameworks are discussed, together with data transport techniques such as XML and SOAP to achieve platform, code and meta-model independent integration.

XML is a meta-markup language for text documents. The data is included as strings of text marked-up by tags describing the data. There are two important features to XML which make it very useful in data transfer [42] [74]:

1. Portability. Just like Java, XML is portable since it is merely a text file and can be directly transferred between platforms. Java and XML produce "portable code, portable data."

2. Inter-operability. The XML standard [127] specifies the format and structure of an XML file but not the content of the tags, the strings, the attributes, etc. An XML file can define airplane data as easily as it can contain a conference paper, as long as it conforms to the formatting standards.

Like XML, SOAP is a standard [128]. SOAP allows for straightforward data transfer using HTTP as the transport layer. The use of HTTP helps to resolve complicated issues as firewalls, ports, sockets, etc.

There are two implementations of the SOAP standard: Apache [113] and Microsoft. The Apache implementation specifies two methods to invoke SOAP services: Remote Process Call and the message-based model. The former is used in this research, and the model can be described with the following steps [112]:

1. A client builds an XML document specifying the server which will service the request, the name of the method and associated parameters used by the method.

2. The server decodes the received XML document and executes the method.

3. After execution the results are packed in a XML document and returned.

4. The client decodes the XML response.

APPENDIX B

AIRCRAFT MODELS

B.1 Aerodynamic Model

Aerodynamic Model Template

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ Preamble
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
ASSIGN FEM=ddelta.fre, PRINT = 0, FORM = FREE, BOUND = ASYM
CEND
$
TITLE = DOUBLE DELTA SBJ
ECHO = SORT
$
SUBCASE =
SUBTITLE =
LABEL =
$
FLUTTER = 10
$
BEGIN BULK
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ Define AERO MODEL
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
AEROZ           NO      NO      KG      M       1.0                     +AE1
+AE1    0.0     0.0     0.0
$
$ FUSELAGE
$
ACOORD  10      0.0     0.0     0.0     0.0     0.0
BODY7   1001    FUSELAGE        10      1       1001
SEGMESH 1001    10      10                                              +SEG11
$               station         radius
+SEG11  1       xxxxxxx         0.0                                     +SEG12
+SEG12  1       xxxxxxx         xxxxxxx                                 +SEG13
+SEG13  1       xxxxxxx         xxxxxxx                                 +SEG14
+SEG14  1       xxxxxxx         xxxxxxx                                 +SEG15
+SEG15  1       xxxxxxx         xxxxxxx                                 +SEG16
+SEG16  1       xxxxxxx         xxxxxxx                                 +SEG17
+SEG17  1       xxxxxxx         xxxxxxx                                 +SEG18
+SEG18  1       xxxxxxx         xxxxxxx                                 +SEG19
+SEG19  1       xxxxxxx         xxxxxxx                                 +SEG20
+SEG20  1       xxxxxxx         0.0                                     +SEG21
$
$ MAIN WING
$                               spandiv chddiv  spandivlist
CAERO7  3001    WING1   10      xxxxxxx xxxxxxx xxxxxxx 0       0       +CA71
$       rootx   rooty   rootz   rootchd chddivlist
+CA71   xxxxxxx xxxxxxx xxxxxxx xxxxxxx xxxxxxx 0                       +CA72
$       tipx    tipy    tipz    tipchd  chddivlist
+CA72   xxxxxxx xxxxxxx xxxxxxx xxxxxxx xxxxxxx 0
CAERO7  4001    WING2   10      xxxxxxx xxxxxxx xxxxxxx 0       0       +CA71
+CA71   xxxxxxx xxxxxxx xxxxxxx xxxxxxx xxxxxxx 0                       +CA72
+CA72   xxxxxxx xxxxxxx xxxxxxx xxxxxxx xxxxxxx 0
CAERO7  5001    WING1R  10      xxxxxxx xxxxxxx xxxxxxx 0       0       +CA71
+CA71   xxxxxxx xxxxxxx xxxxxxx xxxxxxx xxxxxxx 0                       +CA72
+CA72   xxxxxxx xxxxxxx xxxxxxx xxxxxxx xxxxxxx 0
CAERO7  6001    WING2R  10      xxxxxxx xxxxxxx xxxxxxx 0       0       +CA71
+CA71   xxxxxxx xxxxxxx xxxxxxx xxxxxxx xxxxxxx 0                       +CA72
+CA72   xxxxxxx xxxxxxx xxxxxxx xxxxxxx xxxxxxx 0
$
$ HORIZONTAL TAIL
$
CAERO7  7001    HTAIL   10      xxxxxxx xxxxxxx xxxxxxx 0       0       +CA71
+CA71   xxxxxxx xxxxxxx xxxxxxx xxxxxxx xxxxxxx 0                       +CA72
+CA72   xxxxxxx xxxxxxx xxxxxxx xxxxxxx xxxxxxx 0
CAERO7  8001    HTAIL_R 10      xxxxxxx xxxxxxx xxxxxxx 0       0       +CA71
+CA71   xxxxxxx xxxxxxx xxxxxxx xxxxxxx xxxxxxx 0                       +CA72
+CA72   xxxxxxx xxxxxxx xxxxxxx xxxxxxx xxxxxxx 0
$
$ VERTICAL TAIL
$
CAERO7  9001    VTAIL   10      xxxxxxx xxxxxxx xxxxxxx 0       0       +CA71
+CA71   xxxxxxx xxxxxxx xxxxxxx xxxxxxx xxxxxxx 0                       +CA72
+CA72   xxxxxxx xxxxxxx xxxxxxx xxxxxxx xxxxxxx 0
$
```

```
$
$ MKAEROZ TO GENERATE AIC MATRICS FOR MACH AND ANGLE OF ATTACK
$
$               mach    method
MKAEROZ 10      xxxx    x       10      SAVE    AIC_MATRIX              +MK1
+MK1    0.01    0.05    0.1     0.2     0.3     0.4     0.5     0.6     +MK2
+MK2    0.7     0.8     0.9     0.95    0.99    1.0
$                       angle of attack
TRIMFLT 10              xxxxxxx
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ SPLINE OF STRUCTURE TO AERO MODEL
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ FUSELAGE BEAM SPLINE
$
SPLINE2 100             100     10              0.00001
PANLST2 100     1001    1001    THRU    1081
$
$ WING
$
SPLINE1 300                     300     30              0.00001
PANLST1 300     3001    3001    xxxxxxx
SPLINE1 400                     400     40              0.00001
PANLST1 400     4001    4001    xxxxxxx
SPLINE1 500                     500     50              0.00001
PANLST1 500     5001    5001    xxxxxxx
SPLINE1 600                     600     60              0.00001
PANLST1 600     6001    6001    xxxxxxx
$
$ HORIZONTAL TAIL
$
SPLINE1 700                     700     70              0.00001
PANLST2 700     7001    7001    THRU    xxxxxxx
SPLINE1 800                     800     80              0.00001
PANLST2 800     8001    8001    THRU    xxxxxxx
$
$ VERTICAL TAIL
$
SPLINE1 900                     900     90
PANLST2 900     9001    9001    THRU    xxxxxxx
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ FLUTTER ANALYSIS INFO
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$               sym (needs extra space!) nmode
FLUTTER 10      xxxx    20      xxxxxxx  10
TABDMP1 10      G                                                       +TAB1
+TAB1   0.0     0.01    1000.   0.01
$PLTVG   10      10      6       Q               PLTVG.DAT
$
FIXMDEN 20      10      xxxxxxx KG      M       1.0     0       5       +FI1
+FI1    10.     50.     100.    150.    200.    250.    300.    350.    +FI2
+FI2    400.    450.    500.    550.    600.    650.    700.    800.    +FI3
+FI3    1000.   1300.   1600.   2000.   2500.   3100.   3800.   4600.   +FI4
+FI4    5500.
$
$  PLOT AERO MODEL BY PLTAERO
$PLTAERO 11      YES     0       TECPLOT AERO1.PLT
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ IMPORT FROM ANSYS OUT FILE
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ FUSELAGE struct grid point numbers: 10
$ WING struct grid point numbers: 30 40 50 60
$ HORIZONTAL TAIL struct grid point numbers: 70 80
$ VERTICAL TAIL struct grid point numbers: 90
$
$ MAKE THIS ENTIRE STRING IN JAVA AND OUTPUT AT END OF FILE
$
INCLUDE 'aefact.txt'
INCLUDE 'nodes.txt'
ENDDATA
```

B.2 Structural Model

Structural Model Template

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Define the APDL parameters
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
PI          = 3.1415927
!
!!!!!!!!!!!!!!!!!!!
! RANDOM
!!!!!!!!!!!!!!!!!!!
!
!OPTIMIZATION
NMODES      = 4
```

```
w          = 0
DEF_REF    = 1.0
TMASS_REF  = 1.0
!
!MATERIAL
DENSITY   = 2700
YOUNGSMOD = 83000000000
POISSON   = 0.35
!
!MESH DENSITY
MESH_DENSITY_BEAM1   = 2.0
MESH_DENSITY_AREA1   = 2.0
MESH_DENSITY_BEAM2   = 2.0
MESH_DENSITY_AREA2   = 2.0
MESH_DENSITY_CARRY   = 2.0
MESH_DENSITY_FUSE    = 5.0
MESH_DENSITY_BEAM_HT = 2.0
MESH_DENSITY_AREA_HT = 2.0
MESH_DENSITY_BEAM_VT = 2.0
MESH_DENSITY_AREA_VT = 2.0
!
!!!!!!!!!!!!!!!!!!!
! WING
!!!!!!!!!!!!!!!!!!!
!
!GEO Parameters
ROOTX       = 20.0
ROOTY       = 7.0
ROOTZ       = 1.5
SPAN        = 25.0
AR          = 3
TAPER1      = 0.8
TAPER2      = 0.3
TH_TO_CH_1 = 0.12
TH_TO_CH_2 = 0.12
SWEEP1      = 40.0
SWEEP2      = 10.0
NSPAR1      = 3
NRIB1       = 4
NSPAR2      = 3
NRIB2       = 4
!
KINK_LOC   = 0.4
!
!CALCULATED Parameters
SPAN1       = SPAN*KINK_LOC
SPAN2       = SPAN-SPAN1
CHORD_RT    = SPAN1/AR
THICK_RT    = CHORD_RT*TH_TO_CH_1
CHORD_TP1   = CHORD_RT*TAPER1
THICK_TP1   = CHORD_TP1*TH_TO_CH_1
CHORD_TP2   = CHORD_TP1*TAPER2
THICK_TP2   = CHORD_TP2*TH_TO_CH_2
!
!DESIGN Variables
THK_BM1     = 0.10
THK_SH1     = 0.10
THK_BM2     = 0.10
THK_SH2     = 0.10
!
!!!!!!!!!!!!!!!!!!!
! FUSELAGE
!!!!!!!!!!!!!!!!!!!
!
!GEO Parameters
FUSELAGE_LENGTH = 100.0
ENGINE_WEIGHT = 36000.0
!
!DESIGN Variables
THK_CT     = 0.60
THK_FUSE   = 0.10
RAD_FUSE   = 2.00
!
!!!!!!!!!!!!!!!!!!!
!HORIZONTAL TAIL
!!!!!!!!!!!!!!!!!!!
!
!GEO Parameters
ROOTX_HT      = 60.0
ROOTY_HT      = 10.0
ROOTZ_HT      = 2.0
SPAN_HT       = 10.0
AR_HT         = 3
TAPER_HT      = 0.3
TH_TO_CH_HT   = 0.12
SWEEP_HT      = 30.0
NSPAR_HT      = 1
NRIB_HT       = 1
!
!DESIGN Variables
THK_BM_HT     = 0.10
THK_SH_HT     = 0.10
THK_CT_HT     = 0.60
!
!CALCULATED Parameters
CHORD_RT_HT   = SPAN_HT/AR_HT
CHORD_TP_HT   = CHORD_RT_HT*TAPER_HT
THICK_RT_HT   = CHORD_RT_HT*TH_TO_CH_HT
THICK_TP_HT   = CHORD_TP_HT*TH_TO_CH_HT
!
!!!!!!!!!!!!!!!!!!!
```

```
!VERTICAL TAIL
!!!!!!!!!!!!!!!!!!!
!
!GEO_Parameters
ROOTX_VT      = 80.0
ROOTZ_VT      = 5.0
SPAN_VT       = 10.0
AR_VT         = 3
TAPER_VT      = 0.3
TH_TO_CH_VT   = 0.12
SWEEP_VT      = 30.0
NSPAR_VT      = 1
NRIB_VT       = 1
!
!DESIGN_Variables
THK_BM_VT     = THK_BM_HT
THK_SH_VT     = 0.01
THK_CT_VT     = 0.60
!
!CALCULATED Parameters
CHORD_RT_VT   = SPAN_VT/AR_VT
CHORD_TP_VT   = CHORD_RT_VT*TAPER_VT
ROOTY_VT      = CHORD_RT_VT*TH_TO_CH_VT/2
THICK_TP_VT   = CHORD_TP_VT*TH_TO_CH_VT/2
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Define OPTIMIZATION
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
ddelta, , , , , , , , , , , , , , , , , , ,
FINISH
!
*SET, DEF_REF, DEF
*SET, TMASS_REF, TMASS
*SET, OBJFCT, 1.0
!
/OPT
OPANL,'ddelta','mac',' '
!
!
OPVAR,THK_BM1,   DV,0.015,0.700, ,
OPVAR,THK_BM2,   DV,0.015,0.700, ,
OPVAR,THK_SH2,   DV,0.002,0.080, ,
OPVAR,THK_BM_HT, DV,0.015,0.700, ,
!
OPVAR,OBJFCT,OBJ, , , ,
OPDATA, , ,
OPLOOP,PREP,IGNO,SCAL
OPPRNT,OFF
OPKEEP,OFF
!*
!*
OPTYPE,FIRS
OPFRST,5, , ,
!*
!*
OPEXE
!
```

Structural Model Macro

```
!ALL UNITS ARE MKS: METRES, KILOGRAM, SECONDS
!
!
!Preprocessing
/PREP7
*AFUN, DEG
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Generate keypoints WING
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
K,1,ROOTX                                , ROOTY     , ROOTZ,
K,2,ROOTX + CHORD_RT                     , ROOTY     , ROOTZ,
K,3,ROOTX + SPAN1*TAN(SWEEP1) + CHORD_TP1, ROOTY+SPAN1, ROOTZ,
K,4,ROOTX + SPAN1*TAN(SWEEP1)            , ROOTY+SPAN1, ROOTZ,
K,5,ROOTX                                , ROOTY     , ROOTZ+THICK_RT,
K,6,ROOTX + CHORD_RT                     , ROOTY     , ROOTZ+THICK_RT,
K,7,ROOTX + SPAN1*TAN(SWEEP1) + CHORD_TP1, ROOTY+SPAN1, ROOTZ+THICK_TP1,
K,8,ROOTX + SPAN1*TAN(SWEEP1)            , ROOTY+SPAN1, ROOTZ+THICK_TP1,
!
K,9, ROOTX + SPAN1*TAN(SWEEP1) + SPAN2*TAN(SWEEP2) + CHORD_TP2, ROOTY+SPAN1+SPAN2, ROOTZ,
K,10,ROOTX + SPAN1*TAN(SWEEP1) + SPAN2*TAN(SWEEP2), ROOTY+SPAN1+SPAN2 , ROOTZ,
K,11,ROOTX + SPAN1*TAN(SWEEP1) + SPAN2*TAN(SWEEP2) + CHORD_TP2, ROOTY+SPAN1+SPAN2, ROOTZ+THICK_TP2,
K,12,ROOTX + SPAN1*TAN(SWEEP1) + SPAN2*TAN(SWEEP2) , ROOTY+SPAN1+SPAN2, ROOTZ+THICK_TP2,
!
!Make the volume first delta
V,1,2,3,4,5,6,7,8
!
!Make volume second delta
V,4,3,9,10,8,7,11,12
!
! FIRST DELTA
!Move Workplane and create rectangle
!KP for RIBS
*DO,i,1,NRIB1,
```

```
K,20+i-1,ROOTX+SPAN1*TAN(SWEEP1)*i/(NRIB1+1), ROOTY+i*SPAN1/(NRIB1+1), 0,
*ENDDO
!
!KP for SPARS
*DO,i,1,NSPAR1,
K,60+i-1,ROOTX+i*CHORD_RT/(NSPAR1+1), ROOTY, 0,
*ENDDO
!
! SECOND DELTA
!Move Workplane and create rectangle
!KP for RIBS
*DO,i,1,NRIB2,
K,100+i-1,ROOTX+SPAN1*TAN(SWEEP1)+SPAN2*TAN(SWEEP2)*i/(NRIB2+1), ROOTY+SPAN1+i*SPAN2/(NRIB2+1), 0,
*ENDDO
!
!KP for SPARS
*DO,i,1,NSPAR2,
K,140+i-1,ROOTX+SPAN1*TAN(SWEEP1)+i*CHORD_TP1/(NSPAR2+1), ROOTY+SPAN1, 0,
*ENDDO
!
!Define workplane and generate areas for RIBS 1
wpro,,90.0,
*DO,i,1,NRIB1,
KWPAVE,   20+i-1
RECTNG,0, CHORD_RT, 0, ROOTZ+THICK_RT,
*ENDDO
!
!Define workplane and generate areas for SPARS 1
DEL_ANGLE1=(SWEEP1-ATAN((CHORD_TP1-CHORD_RT)/SPAN1+TAN(SWEEP1)))/(NSPAR1+1)
wpro,,-90.0,-90.0
wpro,,-SWEEP1,
*DO,i,1,NSPAR1,
wpro,,+DEL_ANGLE1,
KWPAVE,   60+i-1
RECTNG,0, ROOTZ+THICK_RT, 0, SPAN1*(1+(TAN(SWEEP1-i*DEL_ANGLE1))**2)**0.5,
*ENDDO
!
!Re-init workplane
WPCSYS,,0
!
!Define workplane and generate areas for RIBS 2
wpro,,90.0,
*DO,i,1,NRIB2,
KWPAVE,   100+i-1
RECTNG,0, CHORD_TP1, 0, ROOTZ+THICK_TP1,
*ENDDO
!
!Define workplane and generate areas for SPARS 2
DEL_ANGLE2=(SWEEP2-ATAN((CHORD_TP2-CHORD_TP1)/SPAN2+TAN(SWEEP2)))/(NSPAR2+1)
wpro,,-90.0,-90.0
wpro,,-SWEEP2,
*DO,i,1,NSPAR2,
wpro,,+DEL_ANGLE2,
KWPAVE,   140+i-1
RECTNG,0, ROOTZ+THICK_TP1, 0, SPAN2*(1+(TAN(SWEEP2-i*DEL_ANGLE2))**2)**0.5,
*ENDDO
!
!Re-init workplane
WPCSYS,,0
!
!Select all above areas and divide volume 2
FLST,3,NSPAR2+NRIB2,5,ORDE,2
FITEM,3,(12+NSPAR1+NRIB1)
FITEM,3,-(12+NSPAR1+NRIB1+NSPAR2+NRIB2-1)
VSBA,       2,P51X
!
!Select all above areas and divide volume 1
FLST,3,NSPAR1+NRIB1,5,ORDE,2
FITEM,3,12
FITEM,3,-(12+NSPAR1+NRIB1-1)
VSBA,1,P51X
!
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!HORIZONTAL TAIL KP
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
K,10001,ROOTX_HT                                , ROOTY_HT     , ROOTZ_HT,
K,10002,ROOTX_HT + CHORD_RT_HT                   , ROOTY_HT     , ROOTZ_HT,
K,10003,ROOTX_HT + SPAN_HT*TAN(SWEEP_HT) + CHORD_TP_HT, ROOTY_HT+SPAN_HT, ROOTZ_HT,
K,10004,ROOTX_HT + SPAN_HT*TAN(SWEEP_HT)         , ROOTY_HT+SPAN_HT, ROOTZ_HT,
K,10005,ROOTX_HT                                 , ROOTY_HT     , ROOTZ_HT+THICK_RT_HT,
K,10006,ROOTX_HT + CHORD_RT_HT                   , ROOTY_HT     , ROOTZ_HT+THICK_RT_HT,
K,10007,ROOTX_HT + SPAN_HT*TAN(SWEEP_HT) + CHORD_TP_HT, ROOTY_HT+SPAN_HT, ROOTZ_HT+THICK_TP_HT,
K,10008,ROOTX_HT + SPAN_HT*TAN(SWEEP_HT)         , ROOTY_HT+SPAN_HT, ROOTZ_HT+THICK_TP_HT,
!
!Make the volume
V,10001,10002,10003,10004,10005,10006,10007,10008
!
!Move Workplane and create rectangle
!KP for RIBS
*DO,i,1,NRIB_HT,
K,100020+i-1,ROOTX_HT+SPAN_HT*TAN(SWEEP_HT)*i/(NRIB_HT+1), ROOTY_HT+i*SPAN_HT/(NRIB_HT+1), 0,
*ENDDO
!
!KP for SPARS
*DO,i,1,NSPAR_HT,
K,100060+i-1,ROOTX_HT+i*CHORD_RT_HT/(NSPAR_HT+1), ROOTY_HT, 0,
*ENDDO
```

```
!
!Define workplane and generate areas for RIBS
wpro,,90.0,
*DO,i,1,NRIB_HT,
KWPAVE,   100020+i-1
RECTNG,0, CHORD_RT_HT, ROOTZ_HT+THICK_RT_HT, ROOTZ_HT,
*ENDDO
!
!Define workplane and generate areas for SPARS
DEL_ANGLE_HT=(SWEEP_HT-ATAN((CHORD_TP_HT-CHORD_RT_HT)/SPAN_HT+TAN(SWEEP_HT)))/(NSPAR_HT+1)
wpro,,-90.0,-90.0
wpro,,-SWEEP_HT,
*DO,i,1,NSPAR_HT,
wpro,,+DEL_ANGLE_HT,
KWPAVE,   100060+i-1
RECTNG,ROOTZ_HT+THICK_RT_HT, ROOTZ_HT, 0, SPAN_HT*(1+(TAN(SWEEP_HT-i*DEL_ANGLE_HT))**2)**0.5,
*ENDDO
!
!Re-init workplane
WPCSYS,,0
!
!Select all above areas and divide volume
ASEL,S,LOC,X,ROOTX_HT,ROOTX_HT+SPAN_HT*TAN(SWEEP_HT)+CHORD_TP_HT
ASEL,R,LOC,Y,ROOTY_HT,ROOTY_HT+SPAN_HT
ASEL,R,LOC,Z,ROOTZ_HT,ROOTZ_HT+THICK_RT_HT
*GET,COUNT_MAX, AREA, 0, NUM, MAX
*SET,COUNT,COUNT_MAX
FLST,3,NSPAR_HT+NRIB_HT,5,
FITEM,3,COUNT
*DO,i,1,(NSPAR_HT+NRIB_HT-1),1
*GET,COUNT,AREA,COUNT,NXTL
FITEM,3,COUNT
*ENDDO
VSBA, 1,P51X
!
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!VERTICAL TAIL
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
K,20001,ROOTX_VT                                  , ROOTY_VT            , ROOTZ_VT,
K,20002,ROOTX_VT + CHORD_RT_VT                    , ROOTY_VT            , ROOTZ_VT,
K,20003,ROOTX_VT + SPAN_VT*TAN(SWEEP_VT) + CHORD_TP_VT, THICK_TP_VT     , ROOTZ_VT+SPAN_VT,
K,20004,ROOTX_VT + SPAN_VT*TAN(SWEEP_VT)          , THICK_TP_VT         , ROOTZ_VT+SPAN_VT,
K,20005,ROOTX_VT                                  , -ROOTY_VT           , ROOTZ_VT,
K,20006,ROOTX_VT + CHORD_RT_VT                    , -ROOTY_VT           , ROOTZ_VT,
K,20007,ROOTX_VT + SPAN_VT*TAN(SWEEP_VT) + CHORD_TP_VT, -THICK_TP_VT , ROOTZ_VT+SPAN_VT,
K,20008,ROOTX_VT + SPAN_VT*TAN(SWEEP_VT)          , -THICK_TP_VT        , ROOTZ_VT+SPAN_VT,
!
!Make the volume
V,20001,20002,20003,20004,20005,20006,20007,20008
!
!Move Workplane and create rectangle
!KP for RIBS
*DO,i,1,NRIB_VT,
K,200020+i-1,ROOTX_VT+SPAN_VT*TAN(SWEEP_VT)*i/(NRIB_VT+1), ROOTY_VT, ROOTZ_VT+i*SPAN_VT/(NRIB_VT+1),
*ENDDO
!
!KP for SPARS
*DO,i,1,NSPAR_VT,
K,200060+i-1,ROOTX_VT+i*CHORD_RT_VT/(NSPAR_VT+1), ROOTY_VT, ROOTZ_VT,
*ENDDO
!
!Define workplane and generate areas for RIBS
!wpro,,,
*DO,i,1,NRIB_VT,
KWPAVE,   200020+i-1
RECTNG,0, CHORD_RT_VT, 0, -2*ROOTY_VT,
*ENDDO
!
!Define workplane and generate areas for SPARS
DEL_ANGLE_VT=(SWEEP_VT-ATAN((CHORD_TP_VT-CHORD_RT_VT)/SPAN_VT+TAN(SWEEP_VT)))/(NSPAR_VT+1)
wpro,,,-90.0
wpro,,,SWEEP_VT
*DO,i,1,NSPAR_VT,
wpro,,,-DEL_ANGLE_VT
KWPAVE,   200060+i-1
RECTNG,0, SPAN_VT*(1+(TAN(SWEEP_VT-i*DEL_ANGLE_VT))**2)**0.5, 0, -2*ROOTY_VT
*ENDDO
ALLSEL,ALL
!
!Re-init workplane
WPCSYS,,0
!
!Select all above areas and divide volume
ASEL,S,LOC,X,ROOTX_VT,ROOTX_VT+SPAN_VT*TAN(SWEEP_VT)+CHORD_TP_VT
ASEL,R,LOC,Y,ROOTY_VT,-ROOTY_VT
*GET,COUNT_MAX, AREA, 0, NUM, MAX
*SET,COUNT,COUNT_MAX
FLST,3,NSPAR_VT+NRIB_VT,5,
FITEM,3,COUNT
*DO,i,1,(NSPAR_VT+NRIB_VT-1),1
*GET,COUNT,AREA,COUNT,NXTL
FITEM,3,COUNT
*ENDDO
VSBA, 1,P51X
!
VSEL,ALL
vdele,all
```

```
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!Attach LIFTING SURFACES to FUSELAGE
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
K,1000                  , 0                              , 0, 0,
K,1001                  , ROOTX                          , 0, 0,
!
*DO,i,1,NSPAR1,
K,1002+i-1              , ROOTX+i*CHORD_RT/(NSPAR1+1) , 0, 0,
*ENDDO
!
K,1002+NSPAR1           , ROOTX+CHORD_RT                 , 0, 0,
K,1002+NSPAR1+1         , ROOTX_HT                       , 0, 0,
!
*DO,i,1,NSPAR_HT,
K,1002+NSPAR1+1+i       , ROOTX_HT+i*CHORD_RT_HT/(NSPAR_HT+1), 0, 0,
*ENDDO
!
K,1002+NSPAR1+NSPAR_HT+2, ROOTX_HT+CHORD_RT_HT       , 0, 0,
!
K,1002+NSPAR1+NSPAR_HT+3, ROOTX_VT                    , 0, 0,
!
*DO,i,1,NSPAR_VT,
K,1002+NSPAR1+NSPAR_HT+3+i      , ROOTX_VT+i*CHORD_RT_VT/(NSPAR_VT+1), 0, 0,
*ENDDO
!
K,1002+NSPAR1+NSPAR_HT+NSPAR_VT+4, ROOTX_VT+CHORD_RT_VT, 0, 0,
!
K,1002+NSPAR1+NSPAR_HT+NSPAR_VT+5, FUSELAGE_LENGTH            , 0, 0,
!
! WING TO FUSE
*SET, KP1,KP(ROOTX, ROOTY, ROOTZ)
*SET, KP2,KP(ROOTX, ROOTY, ROOTZ+THICK_RT)
LSTR, KP1, 1001
LSTR, KP2, 1001
*DO,i,1,NSPAR1,
*SET, KP1,KP(ROOTX+i*CHORD_RT/(NSPAR1+1), ROOTY, ROOTZ)
*SET, KP2,KP(ROOTX+i*CHORD_RT/(NSPAR1+1), ROOTY, ROOTZ+THICK_RT)
LSTR, KP1, 1002+i-1
LSTR, KP2, 1002+i-1
*ENDDO
*SET, KP1,KP(ROOTX+CHORD_RT, ROOTY, ROOTZ)
*SET, KP2,KP(ROOTX+CHORD_RT, ROOTY, ROOTZ+THICK_RT)
LSTR, KP1, 1002+NSPAR1
LSTR, KP2, 1002+NSPAR1
!
! HORIZONTAL TAIL TO FUSE
*SET, KP1,KP(ROOTX_HT, ROOTY_HT, ROOTZ_HT)
*SET, KP2,KP(ROOTX_HT, ROOTY_HT, ROOTZ_HT+THICK_RT_HT)
LSTR, KP1, 1002+NSPAR1+1
LSTR, KP2, 1002+NSPAR1+1
*DO,i,1,NSPAR_HT,
*SET, KP1,KP(ROOTX_HT+i*CHORD_RT_HT/(NSPAR_HT+1), ROOTY_HT, ROOTZ_HT)
*SET, KP2,KP(ROOTX_HT+i*CHORD_RT_HT/(NSPAR_HT+1), ROOTY_HT, ROOTZ_HT+THICK_RT_HT)
LSTR, KP1, 1002+NSPAR1+1+i
LSTR, KP2, 1002+NSPAR1+1+i
*ENDDO
*SET, KP1,KP(ROOTX_HT+CHORD_RT_HT, ROOTY_HT, ROOTZ_HT)
*SET, KP2,KP(ROOTX_HT+CHORD_RT_HT, ROOTY_HT, ROOTZ_HT+THICK_RT_HT)
LSTR, KP1, 1002+NSPAR1+NSPAR_HT+2
LSTR, KP2, 1002+NSPAR1+NSPAR_HT+2
!
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!REFLECT the MODEL
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
LSEL,S,LOC,Y,0.01,ROOTY+SPAN1+SPAN2
LSEL,R,LOC,Z,ROOTZ,ROOTZ_HT+THICK_RT_HT
LSYMM,Y,ALL, , , ,1,0
!
ASEL,S,LOC,Y,0.01,ROOTY+SPAN1+SPAN2
ASEL,R,LOC,Z,ROOTZ,ROOTZ_HT+THICK_RT_HT
ARSYM,Y,ALL, , , ,1,0
!
!
! VERTICAL TAIL TO FUSE
*SET, KP1,KP(ROOTX_VT, ROOTY_VT, ROOTZ_VT)
*SET, KP2,KP(ROOTX_VT, -ROOTY_VT, ROOTZ_VT)
LSTR, KP1, 1002+NSPAR1+NSPAR_HT+3
LSTR, KP2, 1002+NSPAR1+NSPAR_HT+3
*DO,i,1,NSPAR_VT,
*SET, KP1,KP(ROOTX_VT+i*CHORD_RT_VT/(NSPAR_VT+1), ROOTY_VT, ROOTZ_VT)
*SET, KP2,KP(ROOTX_VT+i*CHORD_RT_VT/(NSPAR_VT+1), -ROOTY_VT, ROOTZ_VT)
LSTR, KP1, 1002+NSPAR1+NSPAR_HT+3+i
LSTR, KP2, 1002+NSPAR1+NSPAR_HT+3+i
*ENDDO
*SET, KP1,KP(ROOTX_VT+CHORD_RT_VT, ROOTY_VT, ROOTZ_VT)
*SET, KP2,KP(ROOTX_VT+CHORD_RT_VT, -ROOTY_VT, ROOTZ_VT)
LSTR, KP1, 1002+NSPAR1+NSPAR_HT+NSPAR_VT+4
LSTR, KP2, 1002+NSPAR1+NSPAR_HT+NSPAR_VT+4
!
! GET ALL KP ALONG FUSELAGE
KSEL,S,LOC,Y,0
NUMMRG,KP
*GET,NUMKP,KP,0,COUNT
*GET,NXT,KP,0,NUM,MIN
*DIM,KPARRAY,,NUMKP
```

```
*DO,i,1,NUMKP,
   KPARRAY(i)= NXT
    *GET,NXT,KP,NXT,NXTH
*ENDDO
!
!SORT KP BASED ON X LOCATION
*DO,i,1,NUMKP
  *DO,j,1,NUMKP-i
    *GET,LOC1,KP,KPARRAY(j+1),LOC,X
    *GET,LOC0,KP,KPARRAY(j),LOC,X
    *IF,LOC1,LT,LOC0,THEN
         TMP=KPARRAY(j)
         KPARRAY(j)=KPARRAY(j+1)
         KPARRAY(j+1)=TMP
     *ENDIF
  *ENDDO
*ENDDO
!
! CONNECT FUSE KP
*DO,i,1,NUMKP-1,
LSTR, KPARRAY(i), KPARRAY(i+1)
*ENDDO
!
ALLSEL,ALL
NUMMRG,NODE
NUMMRG,KP
!
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!REAL CONSTANTS AND MATERIALS
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
!Assign material properties
MPTEMP,,,,,,,,
MPTEMP,1,0
MPDATA,EX,1,,YOUNGSMOD
MPDATA,PRXY,1,,POISSON
!
!Define types of elements
ET,1,BEAM4
ET,2,SHELL63
!
!Define real constants for:
!WING
R,1,THK_BM1**2,(THK_BM1**4)/12,(THK_BM1**4)/12,THK_BM1,THK_BM1, ,
RMORE, , , , , ,DENSITY*(THK_BM1**2),
R,2,THK_SH1,THK_SH1,THK_SH1,THK_SH1, , ,
RMORE, , , ,DENSITY*THK_SH1,
R,3,THK_BM2**2,(THK_BM2**4)/12,(THK_BM2**4)/12,THK_BM2,THK_BM2, ,
RMORE, , , , , ,DENSITY*(THK_BM2**2),
R,4,THK_SH2,THK_SH2,THK_SH2,THK_SH2, , ,
RMORE, , , ,DENSITY*THK_SH2,
!
! CARRY-THROUGH FUSE
!RECTANGULAR X-SECTION
!R,10,THK_CT**2,(THK_CT**4)/12,(THK_CT**4)/12,THK_CT,THK_CT, ,
!RMORE, , , , ,DENSITY*(THK_CT**2),
!CIRCULAR X-SECTION
R,10,PI*(THK_CT/2)**2,((THK_CT/2)**4)*PI/4,((THK_CT/2)**4)*PI/4,THK_CT,THK_CT, ,
RMORE, , , , , ,DENSITY*(PI*(THK_CT/2)**2),
!
!FUSELAGE
R,20,2*PI*RAD_FUSE*THK_FUSE,(RAD_FUSE**3)*THK_FUSE*PI,(RAD_FUSE**3)*THK_FUSE*PI,RAD_FUSE,RAD_FUSE, ,
RMORE, , , , , ,DENSITY*(2*PI*RAD_FUSE*THK_FUSE),
!
! CARRY-THROUGH HORIZONTAL TAIL
R,30,PI*(THK_CT_HT/2)**2,((THK_CT_HT/2)**4)*PI/4,((THK_CT_HT/2)**4)*PI/4,THK_CT_HT,THK_CT_HT, ,
RMORE, , , , , ,DENSITY*(PI*(THK_CT_HT/2)**2),
!
! HORIZONTAL TAIL
R,40,THK_BM_HT**2,(THK_BM_HT**4)/12,(THK_BM_HT**4)/12,THK_BM_HT,THK_BM_HT, ,
RMORE, , , , , ,DENSITY*(THK_BM_HT**2),
R,50,THK_SH_HT,THK_SH_HT,THK_SH_HT,THK_SH_HT, , ,
RMORE, , , ,DENSITY*THK_SH_HT,
!
!VERTICAL TAIL
R,70,THK_BM_VT**2,(THK_BM_VT**4)/12,(THK_BM_VT**4)/12,THK_BM_VT,THK_BM_VT, ,
RMORE, , , , , ,DENSITY*(THK_BM_VT**2),
R,80,THK_SH_VT,THK_SH_VT,THK_SH_VT,THK_SH_VT, , ,
RMORE, , , ,DENSITY*THK_SH_VT,
!
! CARRY-THROUGH VERTICAL TAIL
R,90,PI*(THK_CT_VT/2)**2,((THK_CT_VT/2)**4)*PI/4,((THK_CT_VT/2)**4)*PI/4,THK_CT_VT,THK_CT_VT, ,
RMORE, , , , , ,DENSITY*(PI*(THK_CT_VT/2)**2),
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! MESHING
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
!MESH FUSELAGE
TYPE,1
MAT, 1
REAL,20
ESYS,0
SECNUM,
ESIZE,MESH_DENSITY_FUSE,0,
MSHAPE,0,2D
MSHKEY,0
LSEL,S,LOC,Y,0.0
```

```
LSEL,R,LOC,Z,0.0
lmesh,all
!
! CARRY-THROUGH FUSELAGE
TYPE,1
MAT, 1
REAL,10
ESYS,0
SECNUM,
ESIZE,MESH_DENSITY_CARRY,0,
MSHAPE,0,2D
MSHKEY,0
LSEL,S,LOC,Y,0.1,ROOTY-0.1
LSEL,R,LOC,X,ROOTX, ROOTX+CHORD_RT
lmesh,all
LSEL,S,LOC,Y,-0.1,-(ROOTY-0.1)
LSEL,R,LOC,X,ROOTX, ROOTX+CHORD_RT
lmesh,all
!
! BEAM 1 WING
TYPE,1
MAT, 1
REAL,1
ESYS,0
SECNUM,
ESIZE,MESH_DENSITY_BEAM1,0,
MSHAPE,0,2D
MSHKEY,0
LSEL,S,LOC,Y,ROOTY,ROOTY+SPAN1
LSEL,R,LOC,X,ROOTX,ROOTX+SPAN1*TAN(SWEEP1)+SPAN2*TAN(SWEEP2)+CHORD_RT
LSEL,R,LOC,Z,ROOTZ,ROOTZ+THICK_RT
lmesh,all
LSEL,S,LOC,Y,-ROOTY,-(ROOTY+SPAN1)
LSEL,R,LOC,X,ROOTX,ROOTX+SPAN1*TAN(SWEEP1)+SPAN2*TAN(SWEEP2)+CHORD_RT
LSEL,R,LOC,Z,ROOTZ,ROOTZ+THICK_RT
lmesh,all
!
! SHELL 1 WING
TYPE,2
MAT, 1
REAL,2
ESYS,0
SECNUM,
ESIZE,MESH_DENSITY_AREA1,0,
MSHAPE,0,2D
MSHKEY,0
ASEL,S,LOC,Y,ROOTY,ROOTY+SPAN1
ASEL,R,LOC,X,ROOTX,ROOTX+SPAN1*TAN(SWEEP1)+SPAN2*TAN(SWEEP2)+CHORD_RT
ASEL,R,LOC,Z,ROOTZ,ROOTZ+THICK_RT
amesh,all
ASEL,S,LOC,Y,-ROOTY,-(ROOTY+SPAN1)
ASEL,R,LOC,X,ROOTX,ROOTX+SPAN1*TAN(SWEEP1)+SPAN2*TAN(SWEEP2)+CHORD_RT
ASEL,R,LOC,Z,ROOTZ,ROOTZ+THICK_RT
amesh,all
!
! BEAM 2 WING
TYPE,1
MAT, 1
REAL,3
ESYS,0
SECNUM,
ESIZE,MESH_DENSITY_BEAM2,0,
MSHAPE,0,2D
MSHKEY,0
LSEL,S,LOC,Y,ROOTY+SPAN1,ROOTY+SPAN1+SPAN2
LSEL,R,LOC,X,ROOTX,ROOTX+SPAN1*TAN(SWEEP1)+SPAN2*TAN(SWEEP2)+CHORD_RT
LSEL,R,LOC,Z,ROOTZ,ROOTZ+THICK_RT
lmesh,all
LSEL,S,LOC,Y,-(ROOTY+SPAN1),-(ROOTY+SPAN1+SPAN2)
LSEL,R,LOC,X,ROOTX,ROOTX+SPAN1*TAN(SWEEP1)+SPAN2*TAN(SWEEP2)+CHORD_RT
LSEL,R,LOC,Z,ROOTZ,ROOTZ+THICK_RT
lmesh,all
!
! SHELL 2 WING
TYPE,2
MAT, 1
REAL,4
ESYS,0
SECNUM,
ESIZE,MESH_DENSITY_AREA2,0,
MSHAPE,0,2D
MSHKEY,0
ASEL,S,LOC,Y,ROOTY+SPAN1,ROOTY+SPAN1+SPAN2
ASEL,R,LOC,X,ROOTX,ROOTX+SPAN1*TAN(SWEEP1)+SPAN2*TAN(SWEEP2)+CHORD_RT
ASEL,R,LOC,Z,ROOTZ,ROOTZ+THICK_RT
amesh,all
ASEL,S,LOC,Y,-(ROOTY+SPAN1),-(ROOTY+SPAN1+SPAN2)
ASEL,R,LOC,X,ROOTX,ROOTX+SPAN1*TAN(SWEEP1)+SPAN2*TAN(SWEEP2)+CHORD_RT
ASEL,R,LOC,Z,ROOTZ,ROOTZ+THICK_RT
amesh,all
!
! CARRY-THROUGH HORIZONTAL TAIL
TYPE,1
MAT, 1
REAL,30
ESYS,0
SECNUM,
ESIZE,MESH_DENSITY_CARRY,0,
MSHAPE,0,2D
MSHKEY,0
```

```
LSEL,S,LOC,Y,0.1,ROOTY_HT-0.1
LSEL,R,LOC,X,ROOTX_HT,ROOTX_HT+CHORD_RT_HT
lmesh,all
LSEL,S,LOC,Y,-0.1,-(ROOTY_HT-0.1)
LSEL,R,LOC,X,ROOTX_HT,ROOTX_HT+CHORD_RT_HT
lmesh,all
!
! BEAM HORIZONTAL TAIL
TYPE,1
MAT, 1
REAL,40
ESYS,0
SECNUM,
ESIZE,MESH_DENSITY_BEAM_HT,0,
MSHAPE,0,2D
MSHKEY,0
LSEL,S,LOC,Y,ROOTY_HT,ROOTY_HT+SPAN_HT
LSEL,R,LOC,X,ROOTX_HT, ROOTX_HT+CHORD_TP_HT+SPAN_HT*TAN(SWEEP_HT)
LSEL,R,LOC,Z,ROOTZ_HT,ROOTZ_HT+THICK_RT_HT
lmesh,all
LSEL,S,LOC,Y,-ROOTY_HT,-(ROOTY_HT+SPAN_HT)
LSEL,R,LOC,X,ROOTX_HT,ROOTX_HT+CHORD_TP_HT+SPAN_HT*TAN(SWEEP_HT)
LSEL,R,LOC,Z,ROOTZ_HT,ROOTZ_HT+THICK_RT_HT
lmesh,all
!
! SHELL HORIZ TAIL
TYPE,2
MAT, 1
REAL,50
ESYS,0
SECNUM,
ESIZE,MESH_DENSITY_AREA_HT,0,
MSHAPE,0,2D
MSHKEY,0
ASEL,S,LOC,Y,ROOTY_HT,ROOTY_HT+SPAN_HT
ASEL,R,LOC,X,ROOTX_HT, ROOTX_HT+CHORD_TP_HT+SPAN_HT*TAN(SWEEP_HT)
ASEL,R,LOC,Z,ROOTZ_HT,ROOTZ_HT+THICK_RT_HT
amesh,all
ASEL,S,LOC,Y,-ROOTY_HT,-(ROOTY_HT+SPAN_HT)
ASEL,R,LOC,X,ROOTX_HT,ROOTX_HT+CHORD_TP_HT+SPAN_HT*TAN(SWEEP_HT)
ASEL,R,LOC,Z,ROOTZ_HT,ROOTZ_HT+THICK_RT_HT
amesh,all
!
! BEAM VERTICAL TAIL
TYPE,1
MAT, 1
REAL,70
ESYS,0
SECNUM,
ESIZE,MESH_DENSITY_AREA_VT,0,
MSHAPE,0,2D
MSHKEY,0
LSEL,S,LOC,Y,ROOTY_VT,-ROOTY_VT
LSEL,R,LOC,Z,ROOTZ_VT, ROOTZ_VT+SPAN_VT
LSEL,R,LOC,X,ROOTX_VT, ROOTX_VT+CHORD_TP_VT+SPAN_VT*TAN(SWEEP_VT)
lmesh,all
!
! SHELL VERTICAL TAIL
TYPE,2
MAT, 1
REAL,80
ESYS,0
SECNUM,
ESIZE,MESH_DENSITY_AREA_VT,0,
MSHAPE,0,2D
MSHKEY,0
ASEL,S,LOC,Y,ROOTY_VT,-ROOTY_VT
ASEL,R,LOC,Z,ROOTZ_VT, ROOTZ_VT+SPAN_VT
ASEL,R,LOC,X,ROOTX_VT, ROOTX_VT+CHORD_TP_VT+SPAN_VT*TAN(SWEEP_VT)
amesh,all
!
! CARRY-THROUGH VERTICAL TAIL
TYPE,1
MAT, 1
REAL,90
ESYS,0
SECNUM,
ESIZE,MESH_DENSITY_CARRY,0,
MSHAPE,0,2D
MSHKEY,0
LSEL,S,LOC,Y,ROOTY_VT,-ROOTY_VT
LSEL,R,LOC,Z,0.1, ROOTZ_VT-0.1
LSEL,R,LOC,X,ROOTX_VT, ROOTX_VT+CHORD_RT_VT
lmesh,all
!
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! OUTPUT NODES CONTROL
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
NSEL,S,LOC,Y,0
/OUTPUT,ansys,txt,
C***, FUSE_10
NLIST,ALL, , ,XYZ,Y,X
/OUTPUT,
! WING1
NSEL,S,LOC,Y,ROOTY,ROOTY+SPAN1
NSEL,R,LOC,Z,ROOTZ
NSEL,R,LOC,X,ROOTX,ROOTX+SPAN1*TAN(SWEEP1)+SPAN2*TAN(SWEEP2)+CHORD_RT
/OUTPUT,ansys,txt,,APPEND
```

```
C***, W1_30
NLIST,ALL, , ,XYZ,Y,X
/OUTPUT,
! WING2
NSEL,S,LOC,Y,ROOTY+SPAN1+0.01,ROOTY+SPAN1+SPAN2
NSEL,R,LOC,Z,ROOTZ
NSEL,R,LOC,X,ROOTX,ROOTX+SPAN1*TAN(SWEEP1)+SPAN2*TAN(SWEEP2)+CHORD_RT
/OUTPUT,ansys,txt,,APPEND
C***, W2_40
NLIST,ALL, , ,XYZ,Y,X
/OUTPUT,
! WING1_R
NSEL,S,LOC,Y,-ROOTY,-(ROOTY+SPAN1)
NSEL,R,LOC,Z,ROOTZ
NSEL,R,LOC,X,ROOTX,ROOTX+SPAN1*TAN(SWEEP1)+SPAN2*TAN(SWEEP2)+CHORD_RT
/OUTPUT,ansys,txt,,APPEND
C***, W1_R_50
NLIST,ALL, , ,XYZ,Y,X
/OUTPUT,
! WING2_R
NSEL,S,LOC,Y,-(ROOTY+SPAN1+0.01),-(ROOTY+SPAN1+SPAN2)
NSEL,R,LOC,Z,ROOTZ
NSEL,R,LOC,X,ROOTX,ROOTX+SPAN1*TAN(SWEEP1)+SPAN2*TAN(SWEEP2)+CHORD_RT
/OUTPUT,ansys,txt,,APPEND
C***, W2_R_60
NLIST,ALL, , ,XYZ,Y,X
/OUTPUT,
! HT
NSEL,S,LOC,Y,ROOTY_HT,ROOTY_HT+SPAN_HT
NSEL,R,LOC,Z,ROOTZ_HT
NSEL,R,LOC,X,ROOTX_HT,ROOTX_HT+SPAN_HT*TAN(SWEEP_HT)+CHORD_RT_HT
/OUTPUT,ansys,txt,,APPEND
C***, HT_70
NLIST,ALL, , ,XYZ,Y,X
/OUTPUT,
! HT_R
NSEL,S,LOC,Y,-ROOTY_HT,-(ROOTY_HT+SPAN_HT)
NSEL,R,LOC,Z,ROOTZ_HT
NSEL,R,LOC,X,ROOTX_HT,ROOTX_HT+SPAN_HT*TAN(SWEEP_HT)+CHORD_RT_HT
/OUTPUT,ansys,txt,,APPEND
C***, HT_R_80
NLIST,ALL, , ,XYZ,Y,X
/OUTPUT,
! VT
NSEL,S,LOC,Z,ROOTZ_VT,ROOTZ_VT+SPAN_VT
NSEL,R,LOC,Y,0,ROOTY_VT
NSEL,R,LOC,X,ROOTX_VT,ROOTX_VT+SPAN_VT*TAN(SWEEP_VT)+CHORD_RT_VT
/OUTPUT,ansys,txt,,APPEND
C***, VT_90
NLIST,ALL, , ,XYZ,Y,X
/OUTPUT,
!
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! ENGINE MASS
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
ET,3,MASS21
R,999, , ,2*ENGINE_WEIGHT, , , ,
!
ALLSEL,ALL
TYPE,3
REAL,999
*SET,ENG_LOC,(ROOTX+CHORD_RT+ROOTX_HT)/2
*SET,ENG_NODE,NODE(ENG_LOC,0,0)
E, ENG_NODE
!
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! SOLUTION CONTROL
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
FINISH
/SOLU
!
!INERTIA RELIEF AND GRAVITY DEF
ACEL,0,0,-9.81,
IRLF,1
!
!Specifiy BOUNDARY CONDITIONS
ALLSEL,ALL
D,1, ,0, , , ,ALL, , , , ,
!
ANTYPE,0
SOLVE
!
*GET,SMASS, ELEM,0,MTOT,X
*GET,MCENT, ELEM,0,MC,X
!
FINISH
/SOLU
!
!Specifies MODAL analysis options
ANTYPE,2
MSAVE,0
MODOPT,LANB,5
EQSLV,SPAR
MXPAND,NMODES, , ,0
```

```
LUMPM,0
PSTRES,0
MODOPT,LANB,NMODES,0, , ,OFF
!
!TAKE INTO ACCOUNT THE WEIGHT OF THE ENGINES
*SET,XC,((MCENT*SMASS+ROOTX_HT*2*ENGINE_WEIGHT)/(SMASS+2*ENGINE_WEIGHT))
*SET,CG,NODE(XC,0,0)
!
!Specifiy BOUNDARY CONDITIONS
DDELE,1,ALL
D,CG, ,0, , , ,ALL, , , , ,
ALLSEL,ALL
!
SOLVE
!
*GET,SMASS, ELEM,0,MTOT,X
*GET,MCENT, ELEM,0,MC,X
*SET,TMASS, SMASS + 2*ENGINE_WEIGHT
!
FINISH
/POST1
!
*SET, DEF, 0
!
*DO,j,1,NMODES,
!
SET,,,,,,,j
AVPRIN,0, ,
!
*DO,i,0,NRIB1+1,
NSEL,S,LOC,Y,ROOTY+i*SPAN1/(NRIB1+1)
NSEL,R,LOC,Z,ROOTZ
NSEL,R,LOC,X,ROOTX,ROOTX+SPAN1*TAN(SWEEP1)+SPAN2*TAN(SWEEP2)+CHORD_RT
*GET,COUNT_MAX, NODE, 0, NUM, MAX
*GET,COUNT_MIN, NODE, 0, NUM, MIN
*GET, DEF1, NODE, COUNT_MIN, U, SUM
*GET, DEF2, NODE, COUNT_MAX, U, SUM
*SET, DEF, DEF+DEF1+DEF2
*ENDDO
!
*DO,i,0,NRIB2+1,
NSEL,S,LOC,Y,ROOTY+SPAN1+i*SPAN2/(NRIB2+1)
NSEL,R,LOC,Z,ROOTZ
NSEL,R,LOC,X,ROOTX,ROOTX+SPAN1*TAN(SWEEP1)+SPAN2*TAN(SWEEP2)+CHORD_RT
*GET,COUNT_MAX, NODE, 0, NUM, MAX
*GET,COUNT_MIN, NODE, 0, NUM, MIN
*GET, DEF1, NODE, COUNT_MIN, U, SUM
*GET, DEF2, NODE, COUNT_MAX, U, SUM
*SET, DEF, DEF+DEF1+DEF2
*ENDDO
!
*DO,i,0,NRIB_HT+1,
NSEL,S,LOC,Y,ROOTY_HT+i*SPAN_HT/(NRIB_HT+1)
NSEL,R,LOC,Z,ROOTZ_HT
NSEL,R,LOC,X,ROOTX_HT,ROOTX_HT+SPAN_HT*TAN(SWEEP_HT)+CHORD_RT_HT
*GET,COUNT_MAX, NODE, 0, NUM, MAX
*GET,COUNT_MIN, NODE, 0, NUM, MIN
*GET, DEF1, NODE, COUNT_MIN, U, SUM
*GET, DEF2, NODE, COUNT_MAX, U, SUM
*SET, DEF, DEF+DEF1+DEF2
*ENDDO
!
*DO,i,0,NRIB_VT+1,
NSEL,S,LOC,Z,ROOTZ_VT+i*SPAN_VT/(NRIB_VT+1)
NSEL,R,LOC,Y,0,ROOTY_VT
NSEL,R,LOC,X,ROOTX_VT,ROOTX_VT+SPAN_VT*TAN(SWEEP_VT)+CHORD_RT_VT
*GET,COUNT_MAX, NODE, 0, NUM, MAX
*GET,COUNT_MIN, NODE, 0, NUM, MIN
*GET, DEF1, NODE, COUNT_MIN, U, SUM
*GET, DEF2, NODE, COUNT_MAX, U, SUM
*SET, DEF, DEF+DEF1+DEF2
*ENDDO
!
*ENDDO
!
C***, MASS IS SET TO:
*SET, TMASS, TMASS
C***, DEFORMATION IS SET TO:
*SET, DEF, DEF
!
*SET, OBJFCT, (1-w)*(DEF/DEF_REF)+w*(TMASS/TMASS_REF)
!
FINISH
!
```

B.3 Performance Model

Performance Model Template

```
 Baseline file for SBJ2003 w/ 2003 Baseline (F110 Der.) engine no TO and L performance

 $OPTION
  IOPT   = 1,
  IANAL  = 3,
  ITAKOF = 0,
```

```
 ILAND  = 0,
 NOISE  = 0,
 ICOST  = 2,
$END

$WTIN
 VMMO   = 2.0,
 DIH    = 7.0,
 FLAPR  = 0.15,
 VARSWP = 0.35,
 NETAW  = 3,
 ETAW   = 0.00, xxxx, 1.00,
 CHD    = xxxx, xxxx, xxxx,
 TOC    = xxxx, xxxx, xxxx,
 SWL    = 00.0, xxxx, xxxx,
 ETAE   = 0.0, 0.0,
 SHT    = xxxx,
 SWPHT  = xxxx,
 ARHT   = xxxx,
 TRHT   = xxxx,
 TCHT   = xxxx,
 HHT    = 1,
 NVERT  = 1,
 SVT    = xxxx,
 SWPVT  = xxxx,
 ARVT   = xxxx,
 TRVT   = xxxx,
 TCVT   = xxxx,
 NFUSE  = 1,
 XL     = xxxx,
 WF     = xxx,
 DF     = xxx,
 XLP    = xxxx,
 NEW    = 0,
 NEF    = 2,
 THRSO  = 38000.0,
 WENG   = 8000,
 XNAC   = 29.2,
 DNAC   = 6.87,
 IFUFU  = 1,
 NPF    = 8,
 NPT    = 0,
 NSTU   = 0,
 NGALC  = 0,
 NFLCR  = 2,
 FRWI   = 0.599,
 FRHT   = 1.37,
 FRVT   = 1.37,
 FRFU   = 0.76,
 FRLGN  = 0.93,
 FRLGM  = 0.93,
 FRNA   = 0,
 WPMSC  = 2,
 WFSYS  = 2,
 WELEC  = 1.4,
 WAVONC = 1.5,
 WFURN  = 1.75,
 WCON   = 0,
$END

$CONFIN
 DESRNG = xxxxxx,
 TWR    = xxxx,
 OFG    = 1,
 OFF    = 0,
 GW     = xxxxxxxx,
 AR     = xxxx,
 SW     = xxxxxx,
 TR     = xxxx,
 SWEEP  = xxxx,
 TCA    = xxxx,
 VCMN   = xxx,
 CH     = xxxxxxxx,
$END

$AERIN
 MYAERO = 1,
 IWAVE  = 1,
 IBO    = 1,
 CLTOM  = 1.0707,
 CLAPP  = 1.154,
 CLLDM  = 1.254,
$END

$ENGDIN
 NGPRT  = 1,
 IGENEN = -1,
 EIFILE = 'baselinedeck',
 IDLE   = 1,
 NONEG  = 1,
 IXTRAP = 1,
 MAXCR  = 1,
 NOX    = 1,
$END

$MISSIN
 FACT   = 1,
 FCDO   = 1,
 FCDI   = 1,
 FCDSUP = 1,
 FCDSUB = 1,
```

```
  ISKAL  = 1,
  IFLAG  = 0,
  MSUMPT = 0,
  IRW    = 1,
  IATA   = 1,
  TAKOTM = 1.,
  TAXOTM = 10.,
  APPRTM = 5.,
  APPFFF = 4.,
  TAXITM = 2.,
  ITTFF  = 1,
  NCLIMB = 2,
  CLMMIN = 0.3, 0.9,
  CLMMAX = 0.9, xxx,
  CLAMIN = 0, 32000.0,
  CLAMAX = 32000.0, xxxxxxxx,
  FWF    = -1, -1,
  NCRCL  = 1, 2,
  NODIVE = 0,
  NCRUSE = 3,
  IOC    = 0, 1, 7
  CRMACH = 0.9, xxx, 0.8
  CRALT  = 32000.0, xxxxxxxx, 20000.0,
  CRMMIN = 0.9, xxx, 0.6,
  CRCLMX = 1.2, 1.2, 1.2,
  IVS    = 1,
  IFAADE = 1,
  DEAMIN = 0.0,
  IRS    = 1,
  TIMMAP = 1.5,
  ALTRAN = 200,
  NCRRES = 3,
  SREALT = 0.0,
  EREALT = 0.0,
  HOLDTM = 30,
  NCRHOL = 3,
  IHOPOS = 1,
  ICRON  = 0,
  THOLD  = 5,
  NCRTH  = 3,
 $END
START
CLIMB    1
CLIMB    2
CRUISE   2
DESCENT
END
  7   7
  0.3000  0.6000  0.9000  1.0000  1.2000  1.6000  2.0000
 0.00000 0.10000 0.20000 0.30000 0.40000 0.50000 0.60000
 0.00000 0.00055 0.00220 0.00494 0.00878 0.01373 0.01976
 0.00000 0.00055 0.00220 0.00494 0.00878 0.01373 0.01976
 0.00000 0.00166 0.00664 0.01494 0.02656 0.04150 0.05976
 0.00000 0.00166 0.00662 0.01490 0.02648 0.04138 0.05958
 0.00000 0.00190 0.00760 0.01711 0.03042 0.04753 0.06844
 0.00000 0.00242 0.00968 0.02179 0.03874 0.06053 0.08716
 0.00000 0.00300 .001199 0.02697 0.04795 0.07493 0.10789
  5   9
  0.0  15000.0  36000.0  50000.0  70000.0
 .3      .6      .9      1.05     1.2    1.4     1.6     1.8     2.0
 0.00669 0.00592 0.00549 0.00525 0.00503 0.00474 0.00447 0.00420 0.00395
 0.00682 0.00604 0.00559 0.00535 0.00512 0.00483 0.00455 0.00428 0.00403
 0.00791 0.00696 0.00642 0.00614 0.00587 0.00554 0.00522 0.00491 0.00462
 0.00880 0.00771 0.00708 0.00677 0.00647 0.00610 0.00575 0.00541 0.00509
 0.01035 0.00900 0.00823 0.00786 0.00751 0.00707 0.00666 0.00626 0.00589
 0.00000 0.00000 0.00000 0.01321 0.00712 0.00651 0.00817 0.00807 0.00679

 $IWGT
  AKRDTE   = 0.,
  AKOANDS  = 0.,
  IPRINTE  = 1,
  IACOST   = 0,
  FENGQ    = 740.,
  PWINGAL  = 1.0,
  PWINGTI  = 0.0,
  PWINGCO  = 0.0,
  PWEMPAL  = 1.0,
  PWEMPTI  = 0.0,
  PWEMPCO  = 0.0,
  PWBODYAL = 1.0,
  PWBODYTI = 0.0,
  PWBODYCO = 0.0,
  PWLGAL   = 1.0,
  PWLGTI   = 0.0,
  PWLGCO   = 0.0,
  PWNACAL  = 1.0,
  PWNACTI  = 0.0,
  PWNACCO  = 0.0,
  $END

 $CMAN
  IRDTE    = 1,
  AGEOI    = 0.2,
  API      = 0.035,
  RTRTN    = 6.00,
  RTRTNA   = 10.0,
  YEAR     = 2003,
  PYEAR    = 2003,
  CFFUSY   = 1.5,
  FEE      = 0.0,
  IAIRPL   = 2,
  IAIRROI  = 1,
```

```
 ICONFG   = 8,
 ICSHFLW  = 1,
 IENGS    = 1,
 IOPS     = 1,
 LEARN1   = 85.0,
 LEARN2   = 85.0,
 LEARNA1  = 85.0,
 LEARNA2  = 85.0,
 LEARNAS1 = 85.0,
 LEARNAS2 = 85.0,
 LEARNFE1 = 82.0,
 LEARNFE2 = 85.0,
 LEARNP1  = 100.0,
 LEARNP2  = 100.0,
 IPROD    = 1,
 PUNITS   = 75.0,
 NV       = 300.0,
 $END

$COPER
 COFL     = 0.71,
 BDMAIN   = 200.0,
 FLF      = 1.0,
 ECLIFE   = 24.0,
 HSUP     = 60000.0,
 HSUB     = 20000.0,
 SL       = 4000.0,
 U        = 1800.0,
 SUBMACH  = 0.90,
 SUBL     = 0.05,
 SUPL     = 0.95,
 RINRST   = 7.0,
 GRNDTM   = 1.0,
 IAFMANT  = 0,
 IEMAINT  = 0,
 $END

$IMAINT
 MTTR     = 1.0,
 MTBF     = 10000.0,
 CF1      = 1.0,
 IMUX     = 1,
 ICIR     = 2,
 IOXG     = 2,
 $END

$RDTE
 PENGMAN  = 0.0,
 CFWAL    = 1.25,
 WLFCLE   = 0.0,
 WLFCMAN  = 0.0,
 WLFCPN   = 0.0,
 WFLTPROV = 228.0,
 WMISPROV = 0.0,
 $END
```

B.4 Others

JAVA Program to Extract Eigenfrequencies

```
/*
 * <code>ddelta_eigen</code> extracts data from an ANSYS/ANSZACON fre file. It extracts
 * the eigenfrequencies to a txt file.
 *
 * Created on October 2003
 */

/**
 *
 * @author  Peter De Baets
 */

import java.io.*;
import java.lang.*;
import java.util.*;

public class ddelta_eigen {

    public static void ansysString(String infilename, String outfilename) {

        String str, string, newString, outString = new String();
        string ="";

        try{
            BufferedReader input = new BufferedReader(new FileReader(infilename));

            outString =  "START \n";
            while((newString = input.readLine())!= null) {

                StringTokenizer token = new StringTokenizer(newString);
                str = (String)token.nextToken();

                if(str.equals("$")){
                    token.nextToken();
                    str = (String)token.nextToken();
                    int count = Integer.valueOf(str).intValue();
```

```
                    newString = input.readLine();
                    token = new StringTokenizer(newString);
                    str = (String)token.nextToken();
                    double eigenfreq = Double.valueOf(str).doubleValue();
                    outString =  outString + count + " " + eigenfreq + "\n";
                }

            }
            outString = outString + "END";
            input.close();
        } catch (IOException e) {System.err.println("Unable to read from file");}

        makeFile(outfilename, outString);
    }

    public static void makeFile(String outfilename, String str){
        try{
            PrintWriter output = new PrintWriter( new BufferedWriter( new FileWriter(outfilename)));

            output.print(str);
            output.close();
        } catch (IOException e) {System.err.println("Unable to write to file");}
    }

    public static void main(String[] args) {
        ansysString("file.fre", "freq.txt");
    }
}
```

JAVA Program to Extract Eigenmodes

```
/*
 * <code>ddelta_mode</code> extracts the eigenmodes from an ANSYS/ANSZACON free format and
 * nodes.txt file. It sorts them in a ZAERO readable free format.
 *
 * Created on August 2003
 */

/**
 *
 * @author  Peter De Baets
 */

import java.io.*;
import java.lang.*;
import java.util.*;

public class ddelta_mode {

    public static void ansysString(String infilename, String frefilename, String outfilename) {
        int nodeNumber;
        String assemblyPart;
        String output = "";

        String string, newString = new String();
        string ="";
        try{

            BufferedReader input = new BufferedReader(new FileReader(infilename));
            while((newString = input.readLine())!= null) {
                if (newString.equals("1")){
                    string = string + " cont ";
                } else {
                    string = string + newString;
                }
            }

            input.close();
        } catch (IOException e) {System.err.println("Unable to read from file");}

        boolean loopEnd = true;
        boolean nodeEnd = true;
        boolean nodeContinue = true;
        boolean nodeExtended = true;
        String[] setArray = new String[10000];
        int i = 0;

        StringTokenizer token = new StringTokenizer(string);
        String str;
        str = (String)token.nextToken();

        while(token.hasMoreTokens() & nodeEnd){
            if(str.equals("C***,")) {
                token.nextToken();
                while(token.hasMoreTokens() & loopEnd & nodeEnd){
                    if(str.equals("NODE")) {
                        while(token.hasMoreTokens() & loopEnd & nodeContinue &
                                                                token.countTokens() > 4 ){

                            if(str.equals("C***,")) {
                                loopEnd = false;
                            } else {
                                token.nextToken();
                                token.nextToken();
                                token.nextToken();
```

```
                                str = (String)token.nextToken();
                                if(!(str.equals("C***,"))) {
                                    if(str.equals("cont")) {
                                        str = (String)token.nextToken();
                                        if(str.equals("*****")) {
                                            while(!(str.equals("NODE"))){
                                                str = (String)token.nextToken();
                                            }
                                            nodeExtended = false;
                                        } else {
                                            nodeContinue = false;
                                        }
                                    } else {
                                        nodeNumber =  Integer.valueOf(str).intValue();
                                        setArray[i] = str;
                                        i++;
                                    }
                                }
                            }
                        }
                        nodeContinue = true;
                        nodeExtended = true;
                        if(token.countTokens() < 4){ nodeEnd = false;}
                    } else {
                        str = (String)token.nextToken();
                    }
                }
            }
            loopEnd = true;
        }
        deleteLinesFre(frefilename, outfilename, setArray, i);
    }

    public static void deleteLinesFre(String frefilename, String outfilename, String[] setArray,
                                                                                    int setLength){

        String string, newString = new String();
        string = "";

        try{
            BufferedReader input = new BufferedReader(new FileReader( frefilename));

            input.readLine();
            string = Integer.toString(setLength) + " 55 \n";
            while((newString = input.readLine())!= null) {
                //System.out.println(i);
                StringTokenizer token = new StringTokenizer(newString);
                String str = (String)token.nextToken();

                switch(str.charAt(0)){
                    case '$':
                        string = string + newString + " \n";
                        newString = input.readLine();
                        string = string + newString + " \n";
                        break;
                    default:
                        for(int j = 0; j < setLength; j++){
                            if (str.equals(setArray[j].toString())){
                                string = string + newString + " \n";
                                j = setLength;
                            }
                        }
                        break;
                }
            }
            input.close();

            PrintWriter output = new PrintWriter( new BufferedWriter( new FileWriter( outfilename)));
            output.print(string);
            output.close();
        } catch (IOException e) {System.err.println("Unable to write to file");}

    }

    public static void main(String[] args) {
        ansysString("ansys.txt", "file.fre", "ddelta.fre.template");
    }
}
```

APPENDIX C

VALIDATION AND CHECKS RESULTS

C.1 Global Design Variable Screening

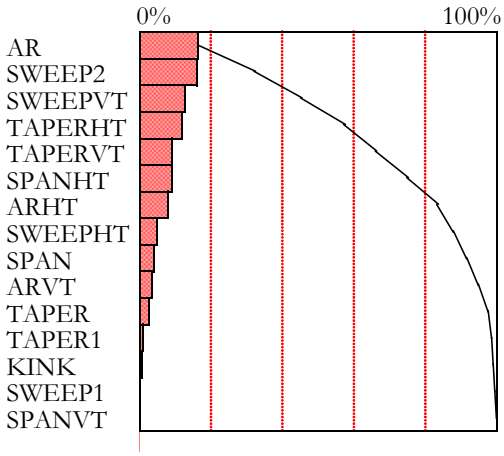

Figure 97: Effect of Global Geometric Design Variables on Divergence Speed Variability

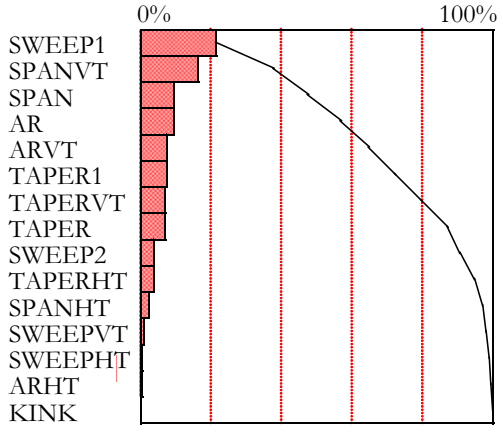

Figure 98: Effect of Global Geometric Design Variables on Flutter Speed Variability

C.2 Local Design Variable Screening

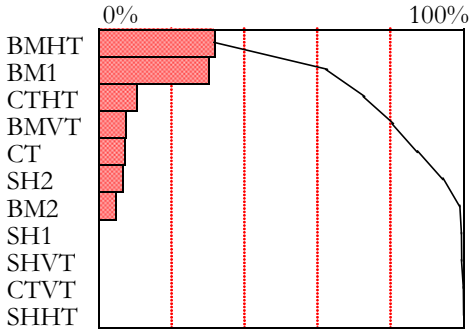

Figure 99: Effect of ANSYS Local Design Variables on Divergence Speed Variability

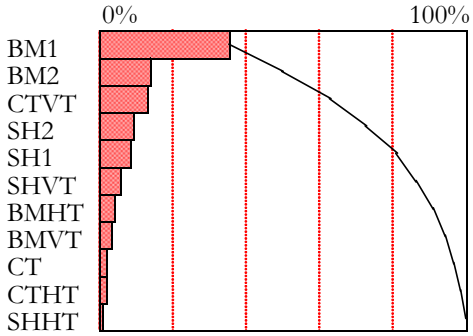

Figure 100: Effect of ANSYS Local Design Variables on Flutter Speed Variability

C.3 Aerodynamic Mesh Density

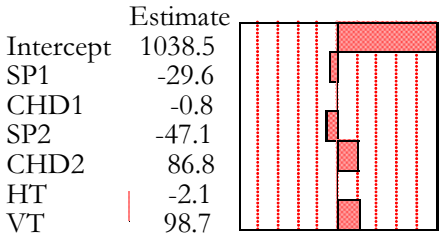

Figure 101: Scaled Estimate of ZAERO Mesh Density Changes on Divergence Speed Variability

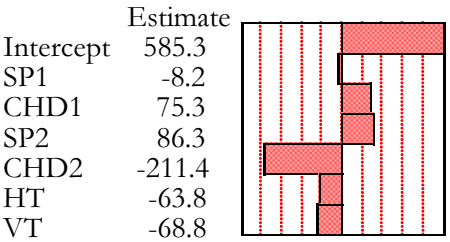

Figure 102: Scaled Estimate of ZAERO Mesh Density Changes on Flutter Speed Variability

C.4 Structural Mesh Density

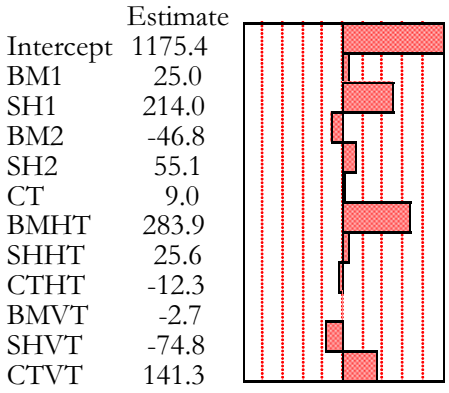

Figure 103: Scaled Estimate of ANSYS Mesh Density Changes on Divergence Speed Variability

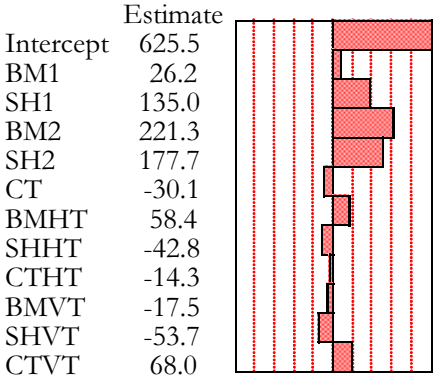

Figure 104: Scaled Estimate of ANSYS Mesh Density Changes on Flutter Speed Variability

C.5 Effect of Number of Eigenmodes on Aeroelastic Speeds

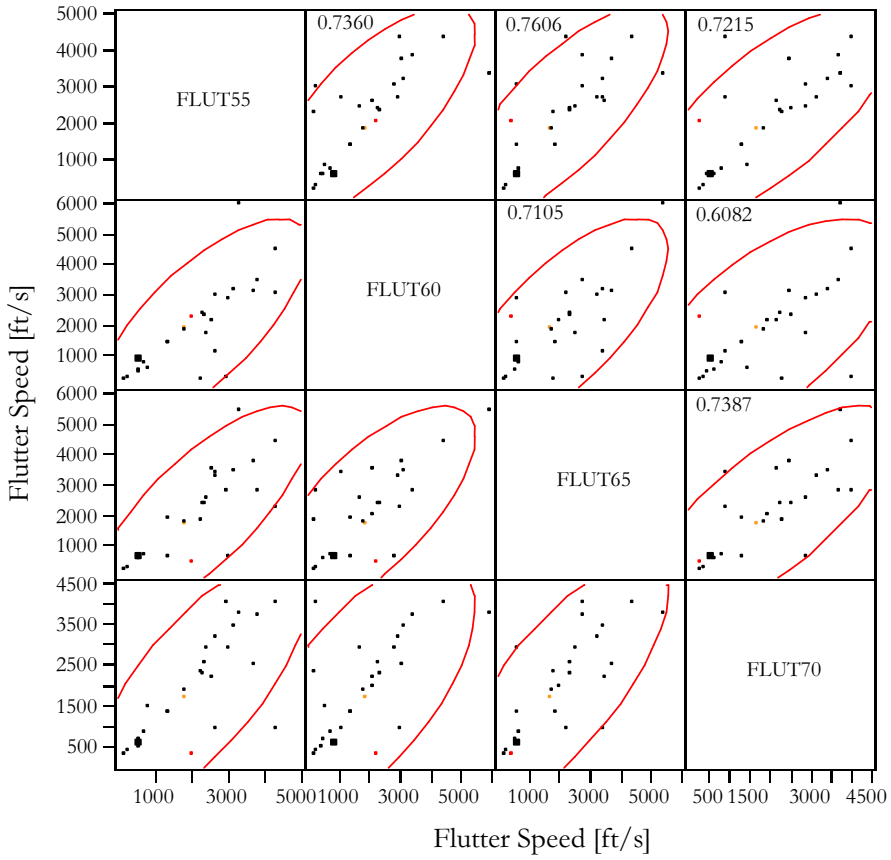

Figure 105: Correlation of Flutter Speeds with Increasing Number of Eigenmodes

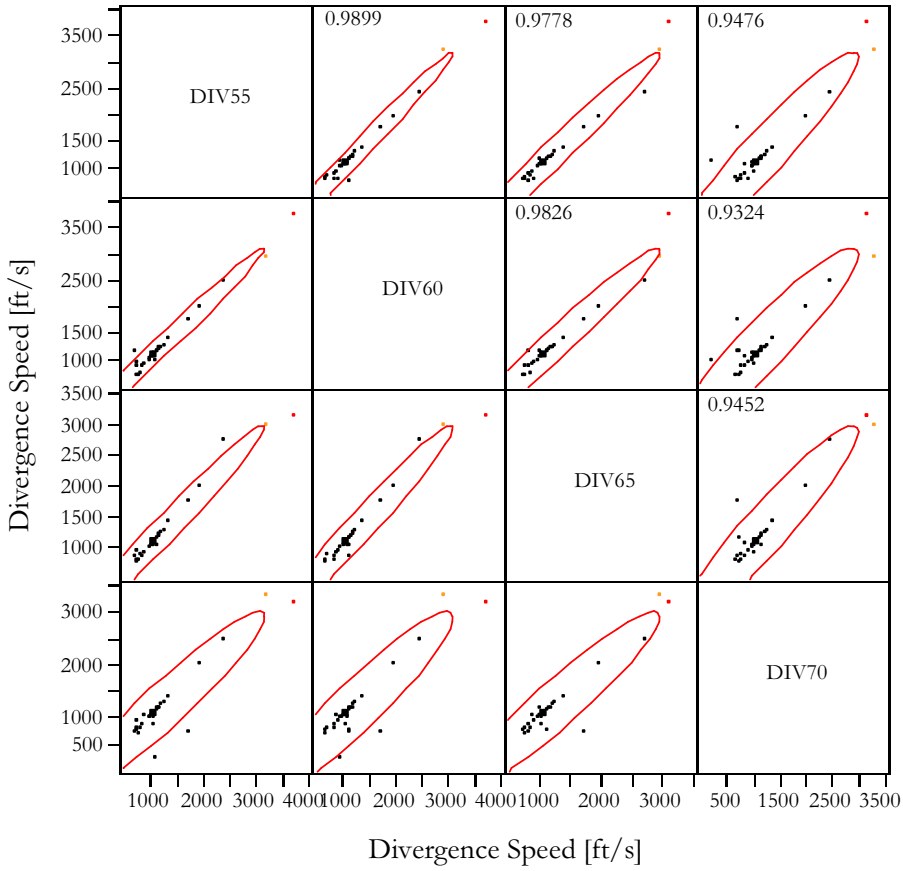

Figure 106: Correlation of Divergence Speeds with Increasing Number of Eigenmodes

C.6 Correlation of Eigenmodes

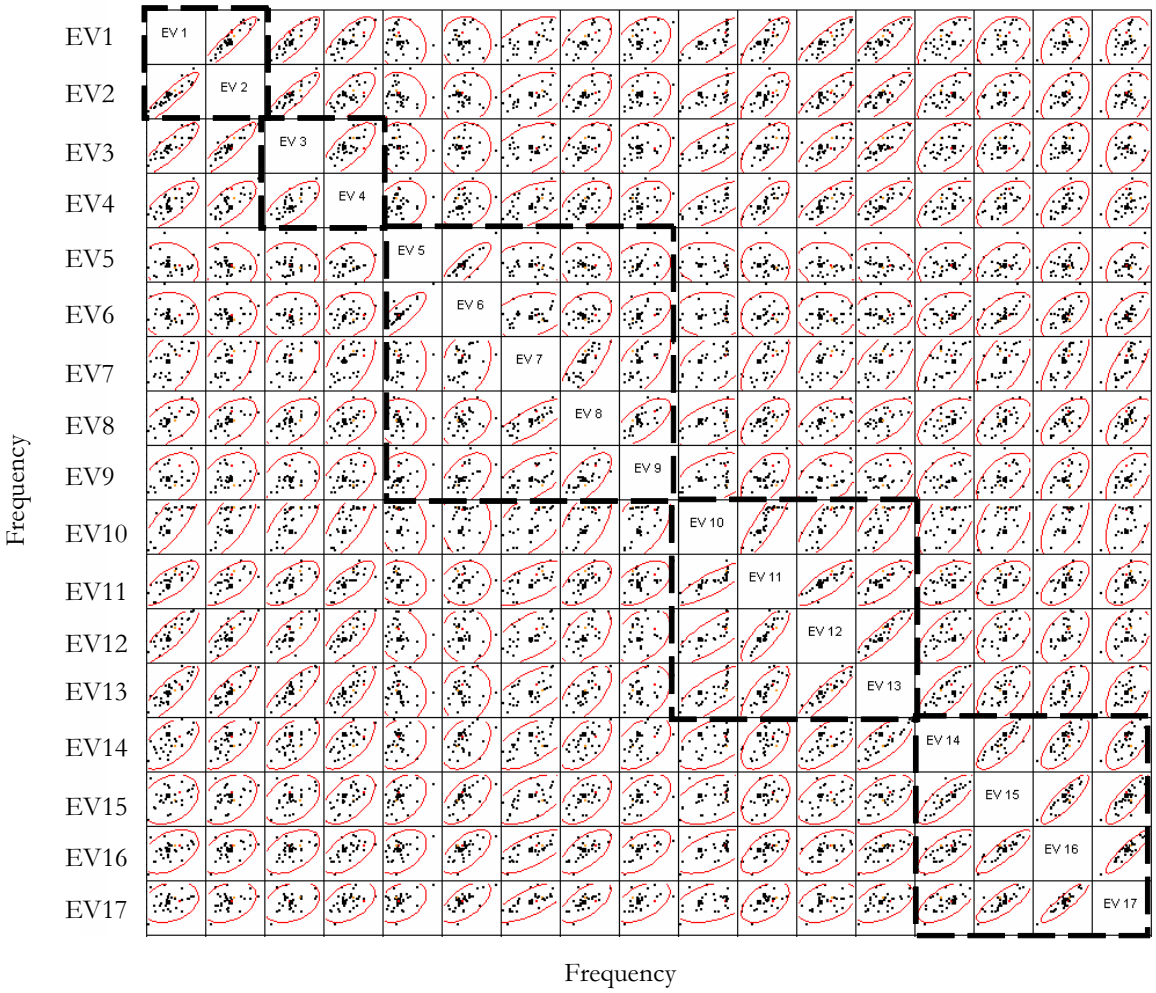

Figure 107: Correlation of Eigenvalues 1 Through 17 with Change of Global Design Variables

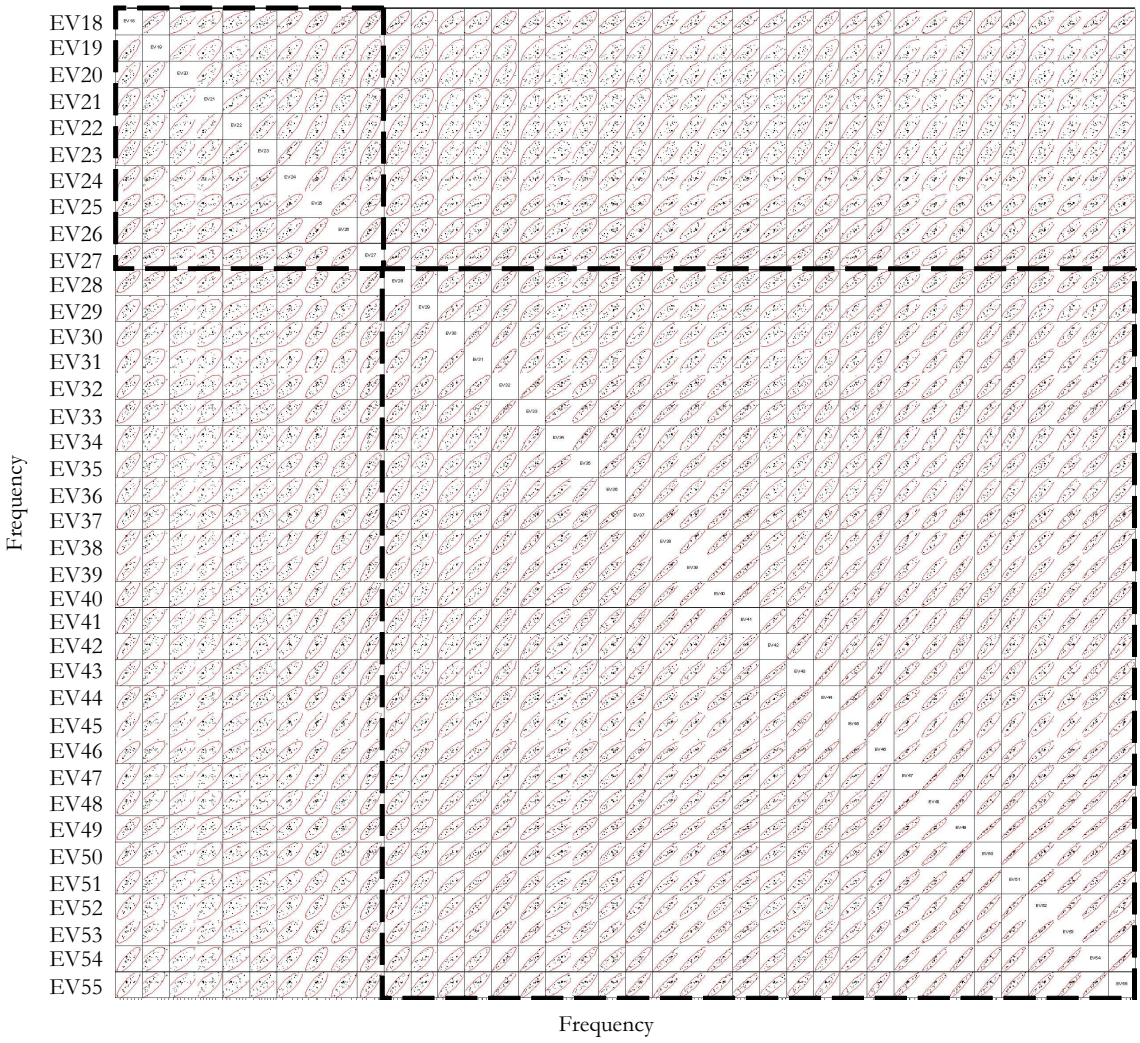

Figure 108: Correlation of Eigenvalues 18 through 55 with Change of Global Design Variables

C.7 Impact of Eigenmodes on Variability of Aeroelastic Speeds

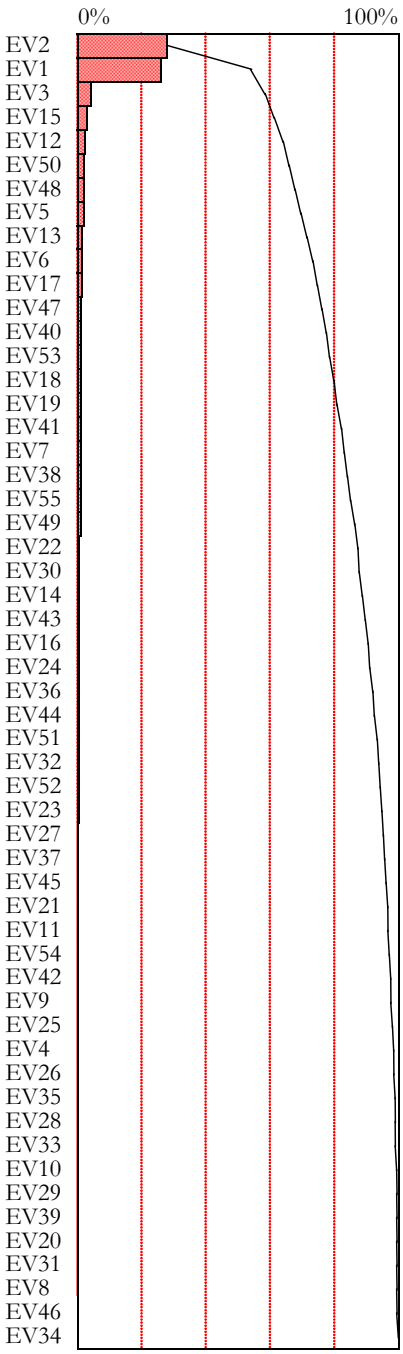

Figure 109: Impact of Eigenvalues on Flutter Speed Speed Variability

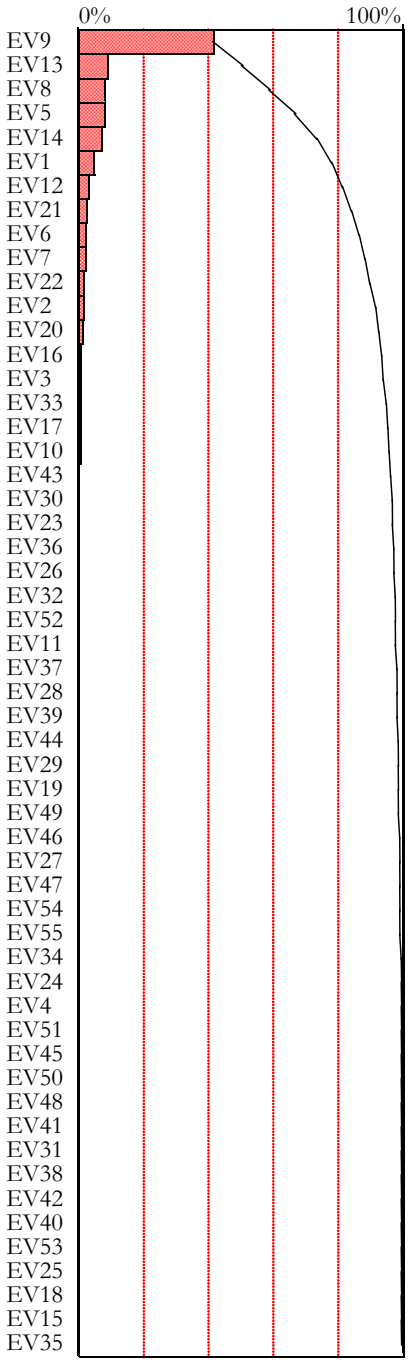

Figure 110: Impact of Eigenvalues on Divergence Speed Variability

APPENDIX D

BLISS OPTIMIZATION RESULTS

D.1 First Optimization Run

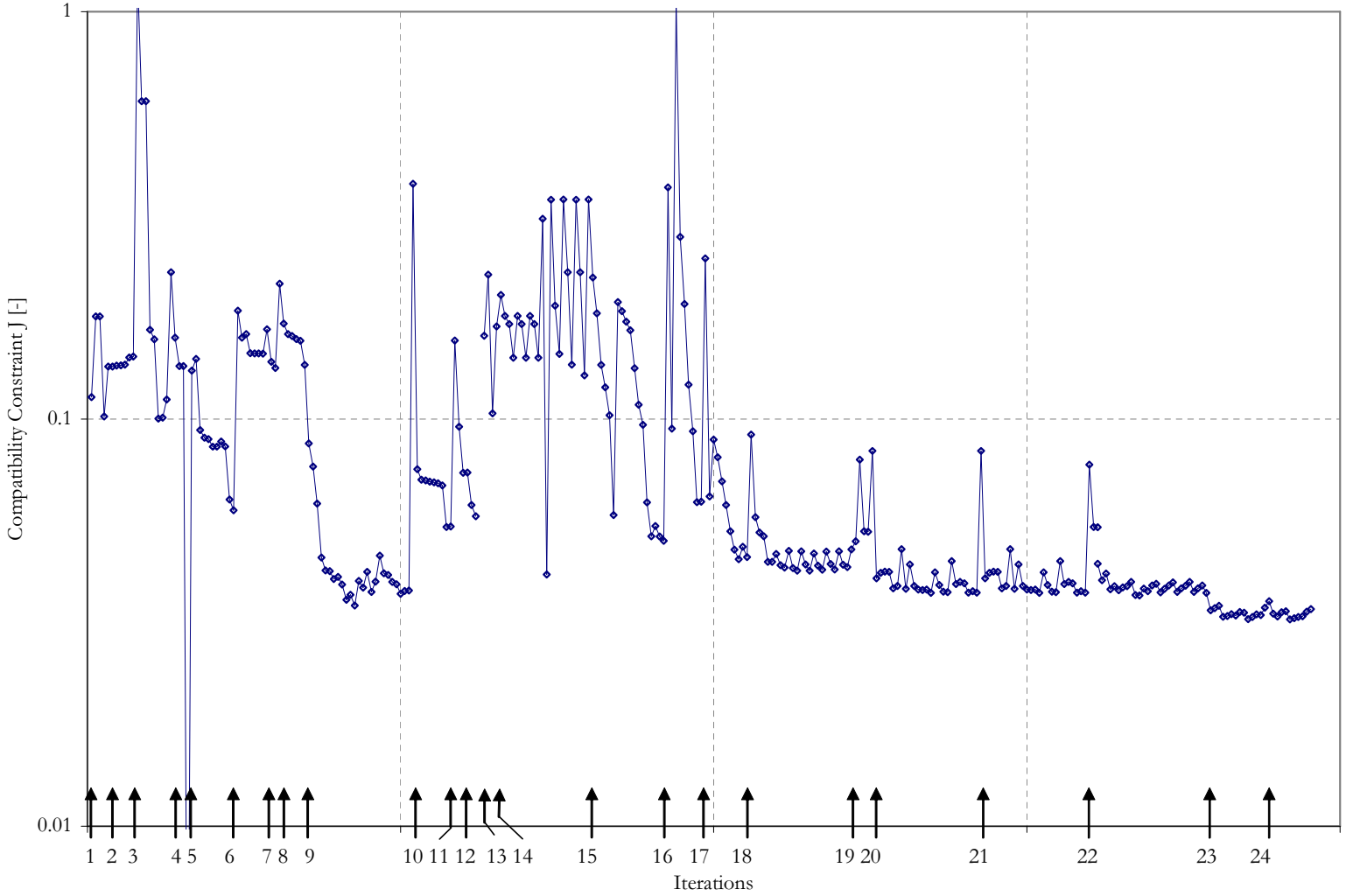

Figure 111: First Vehicle - Compatibility Constraint J

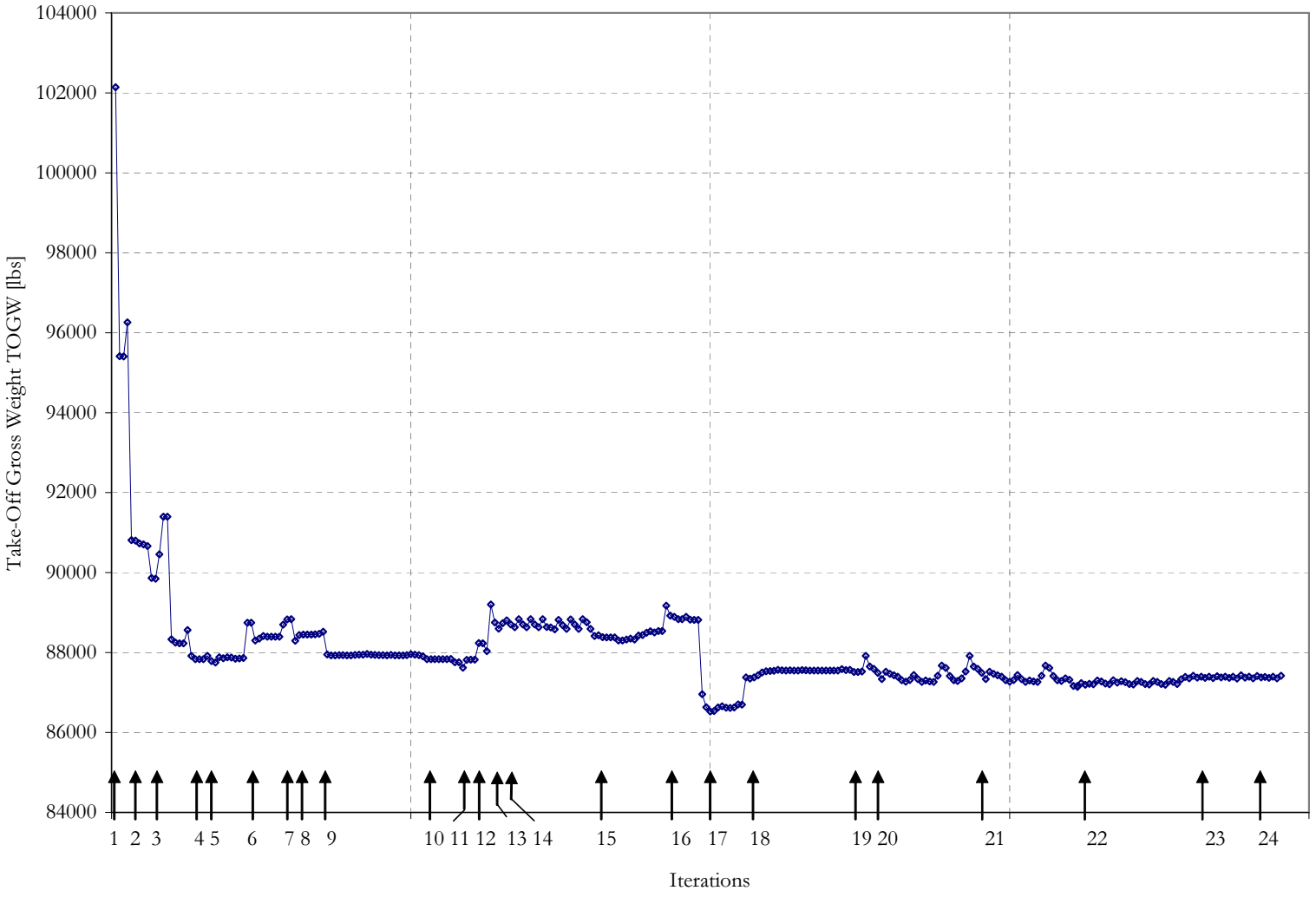

Figure 112: First Vehicle - Take-off Gross Weight

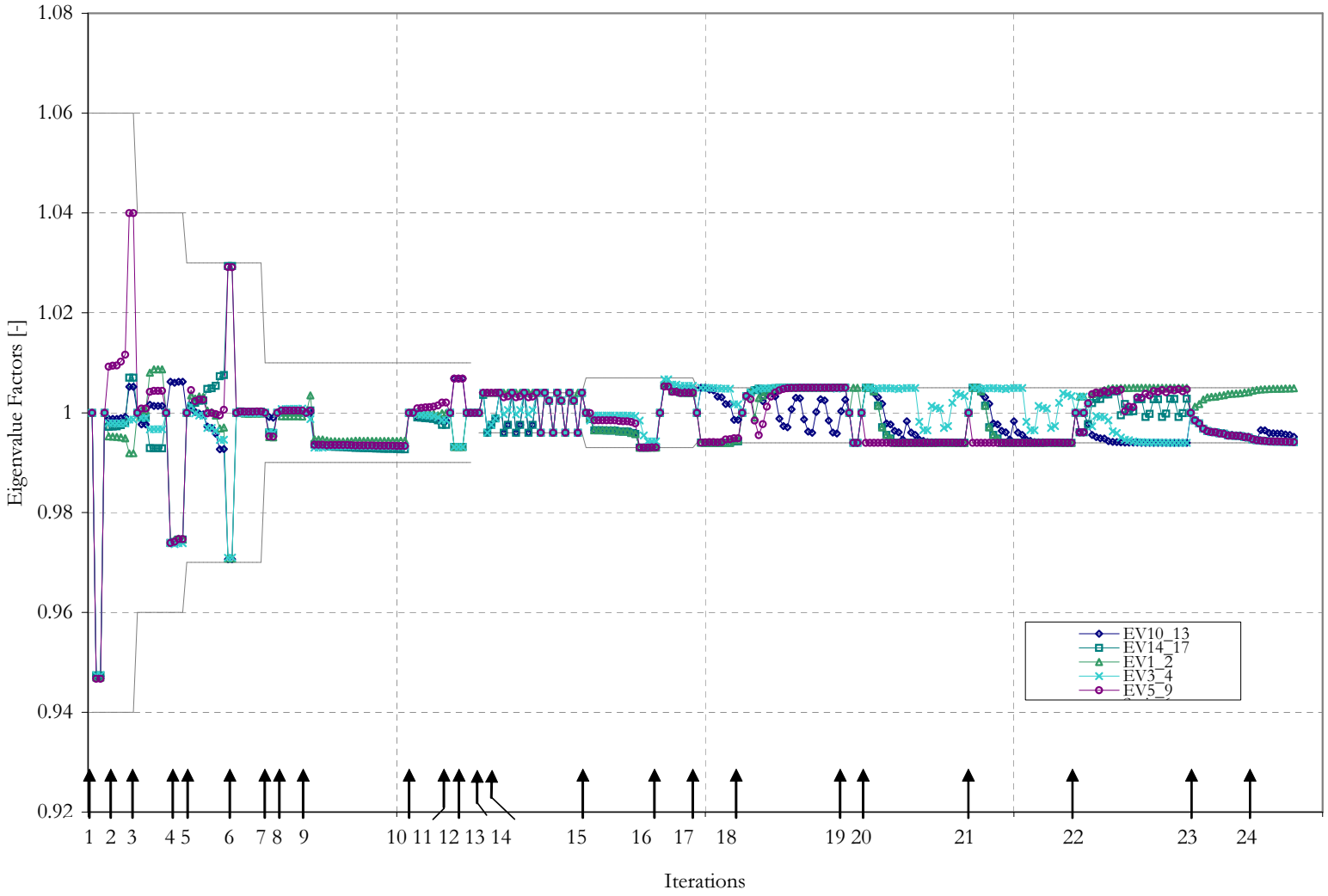

Figure 113: First Vehicle - Eigenvalue Pooling Factors

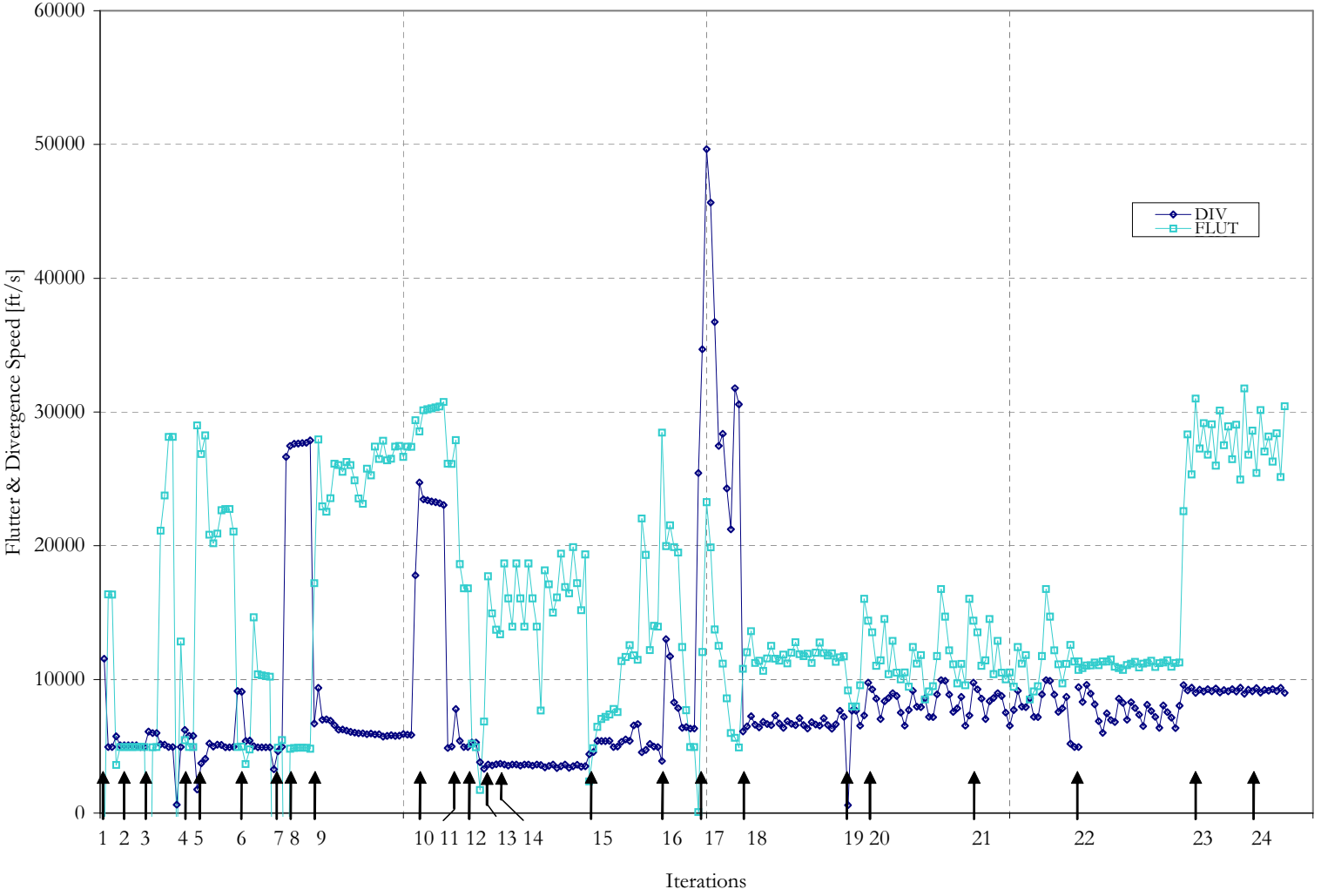

Figure 114: First Vehicle - Flutter and Divergence Speed

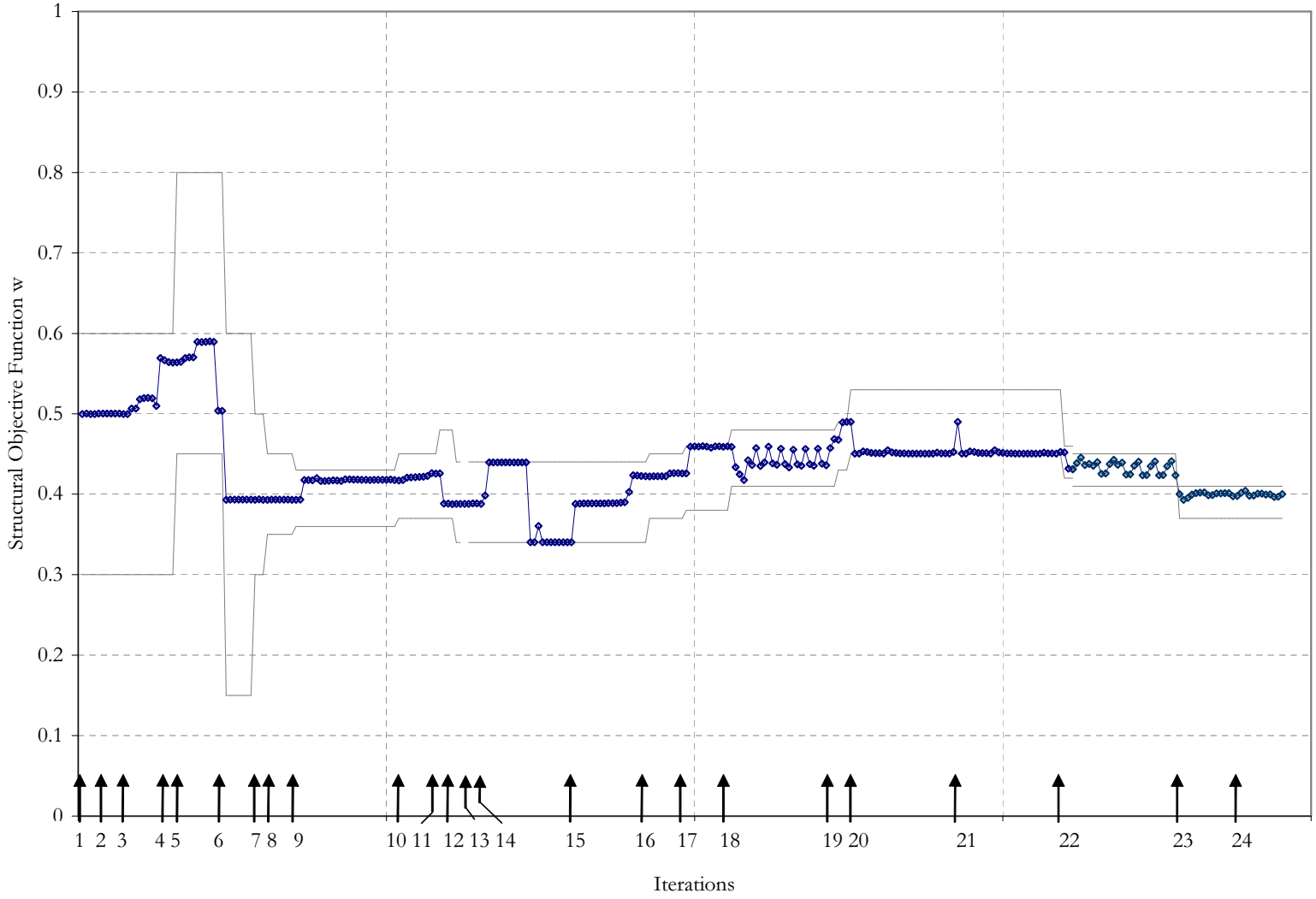

Figure 115: First Vehicle - Structural Objective Weight w

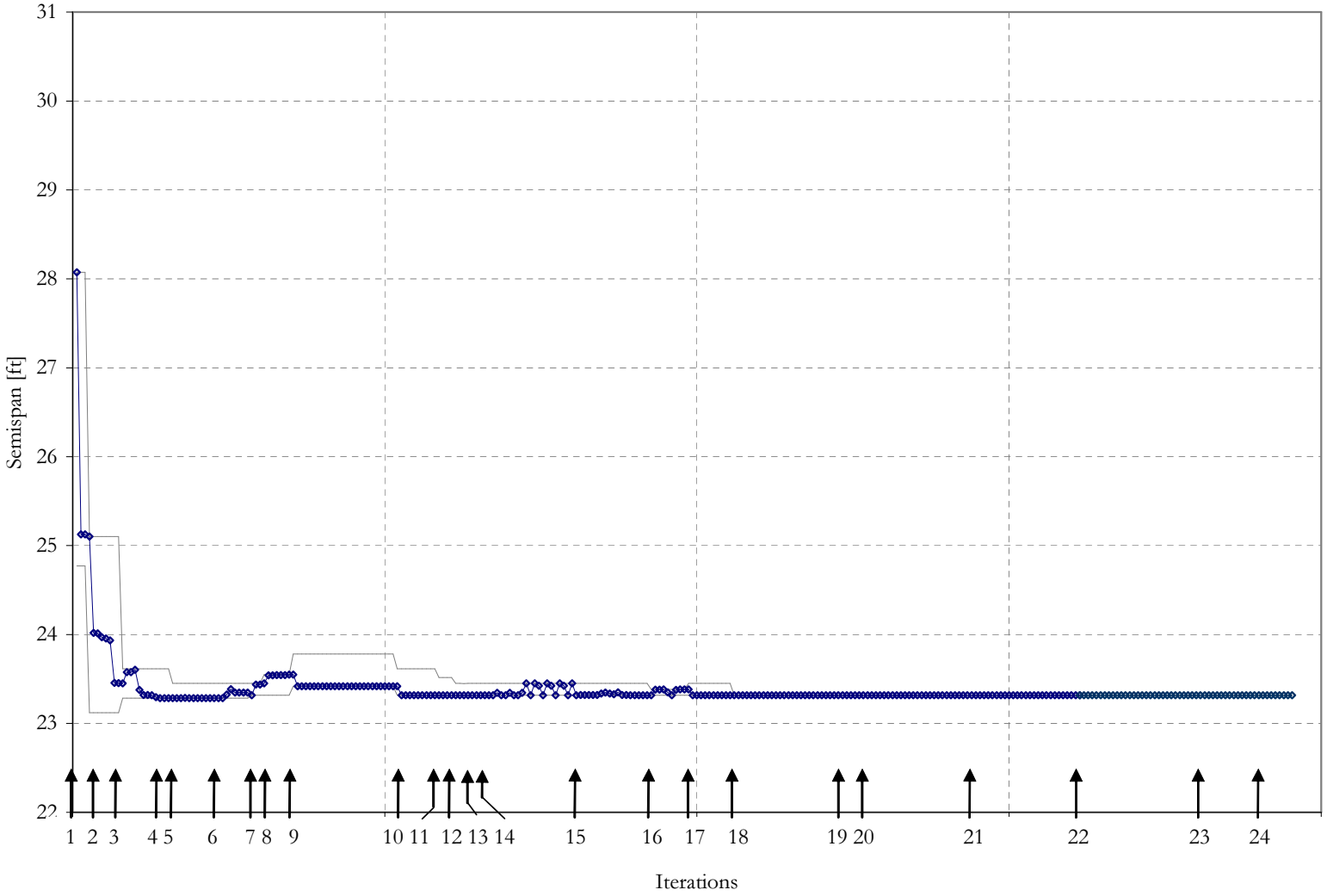

Figure 116: First Vehicle - Geometric Variables, Semi-span

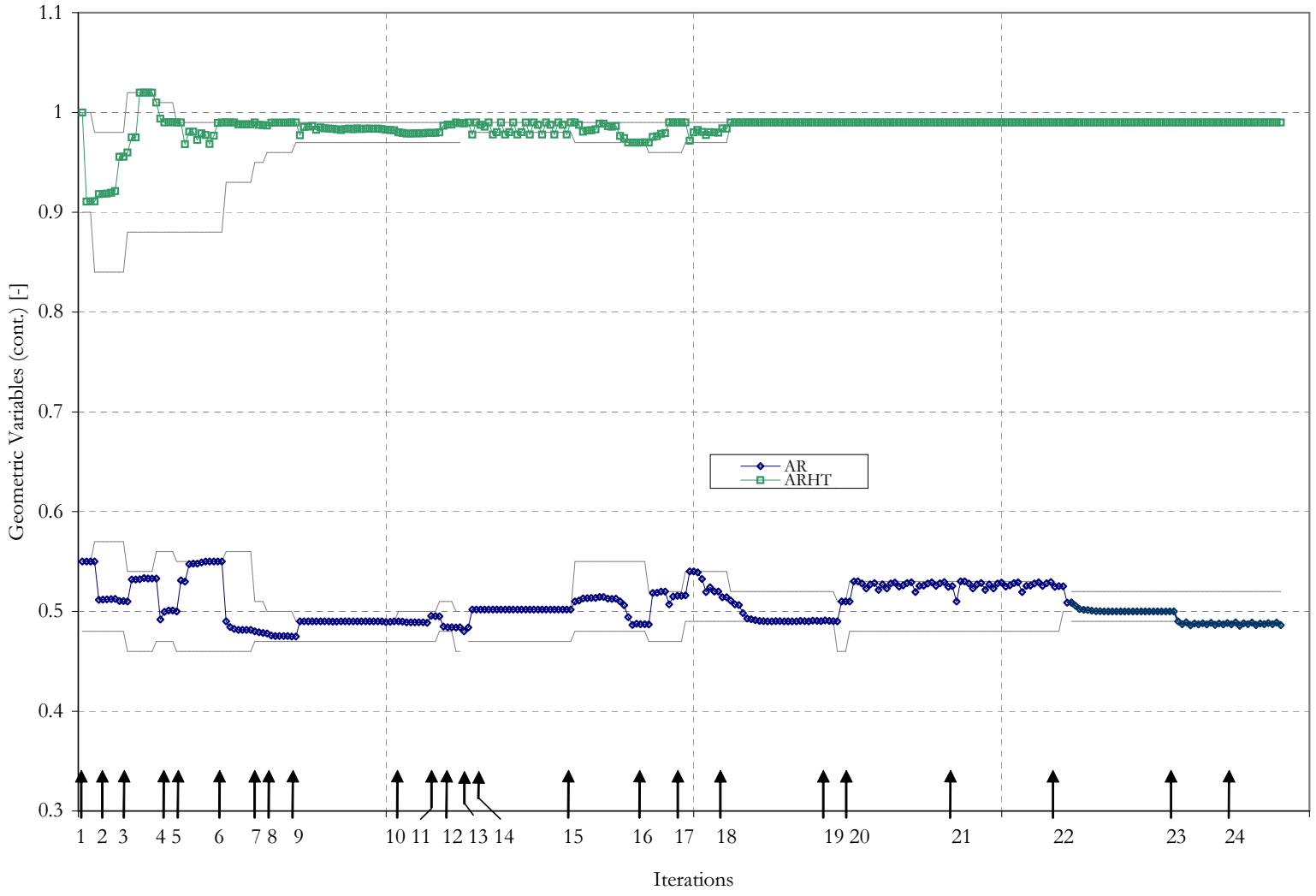

Figure 117: First Vehicle - Geometric Variables, Aspect Ratio

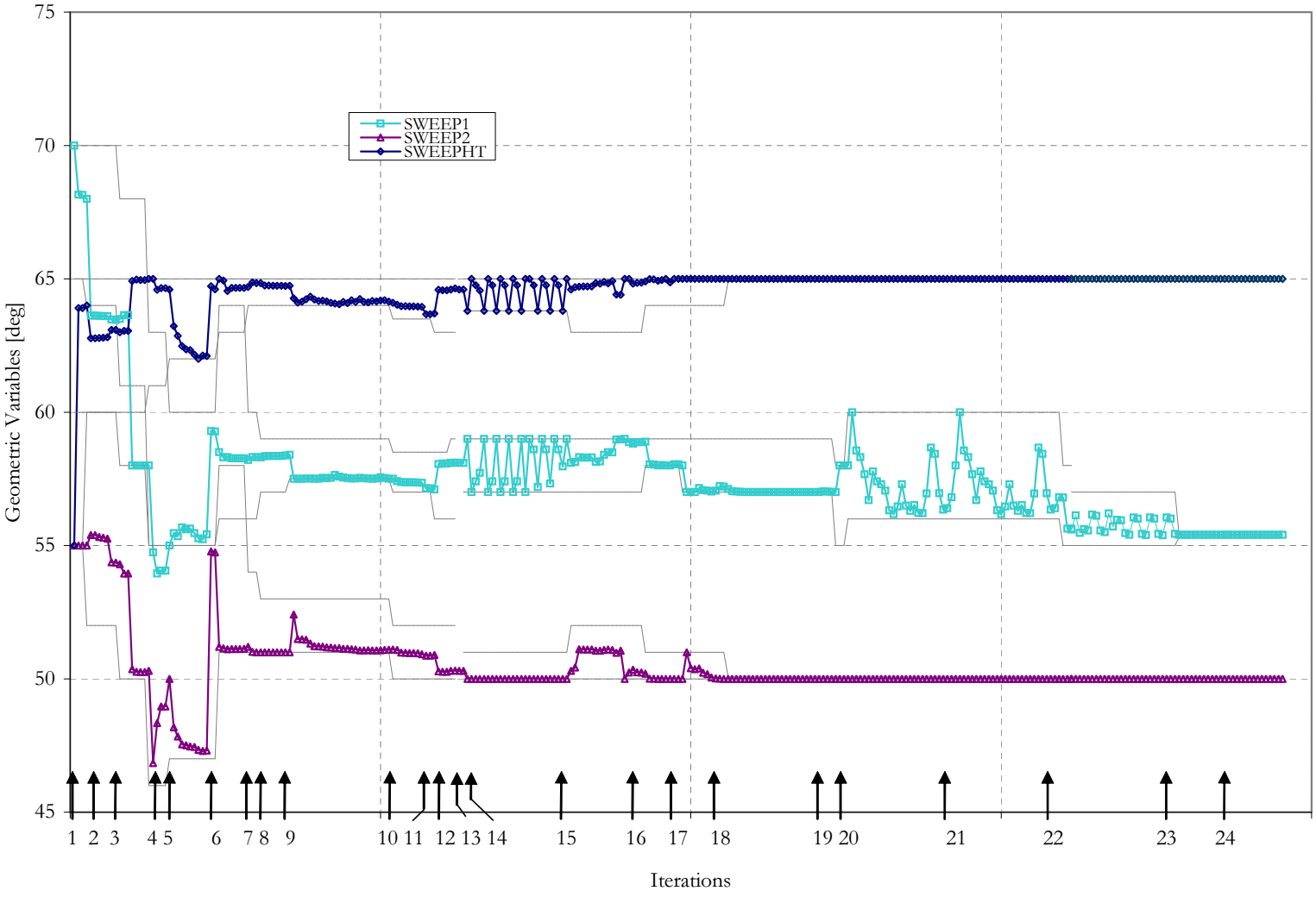

Figure 118: First Vehicle - Geometric Variables, Sweep

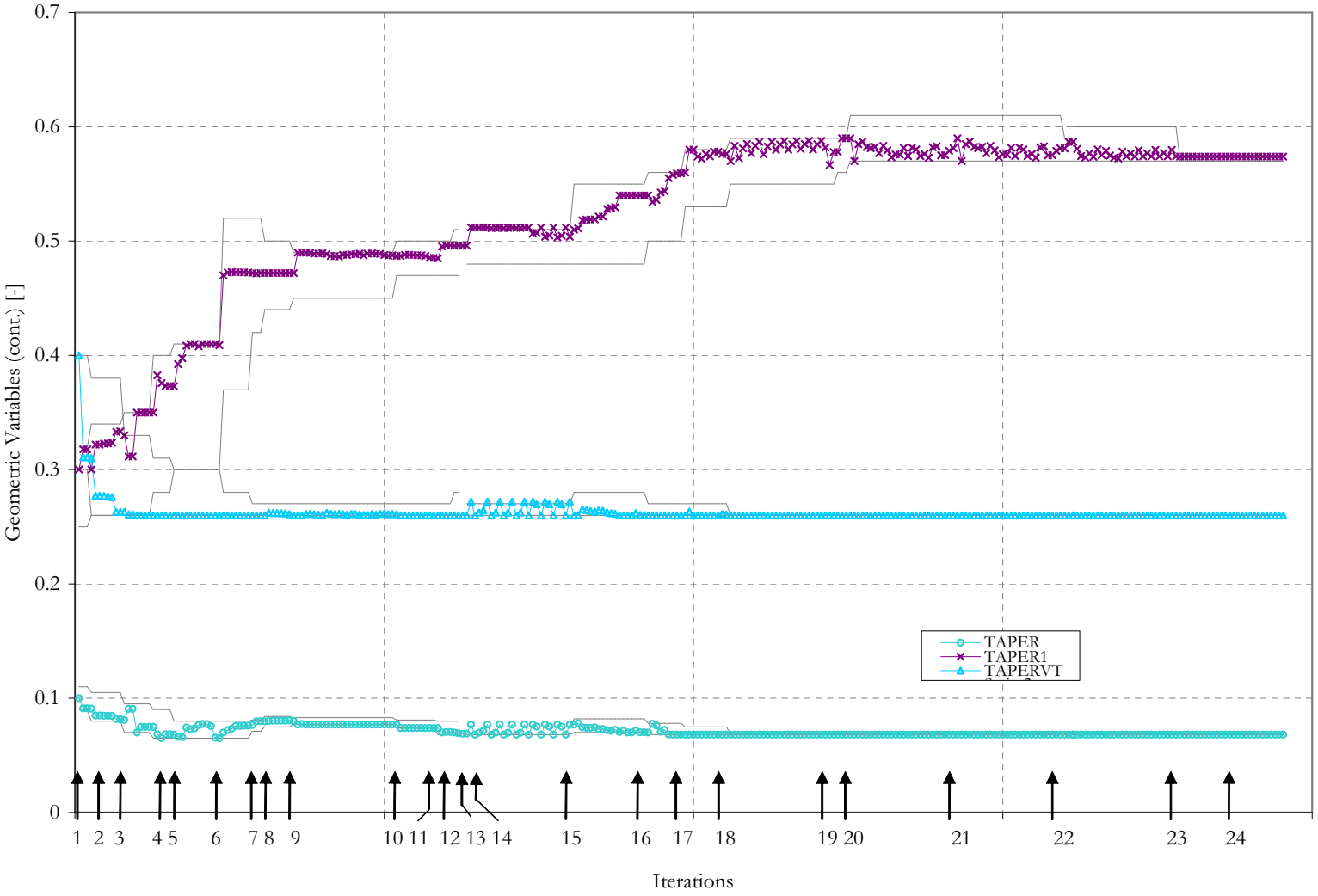

Figure 119: First Vehicle - Geometric Variables, Taper

D.2 Second Optimization Run

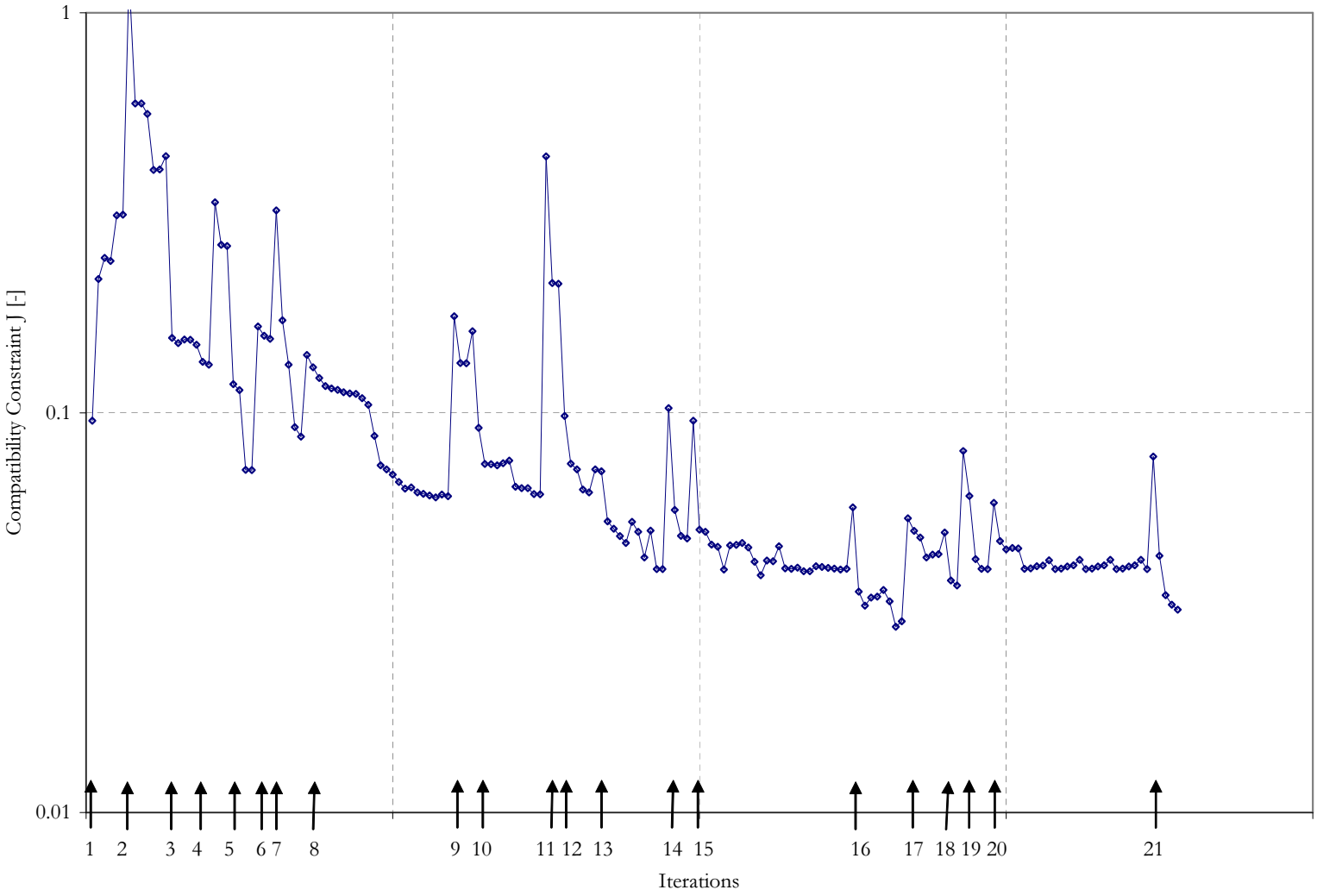

Figure 120: Second Vehicle - Compatibility Constraint J

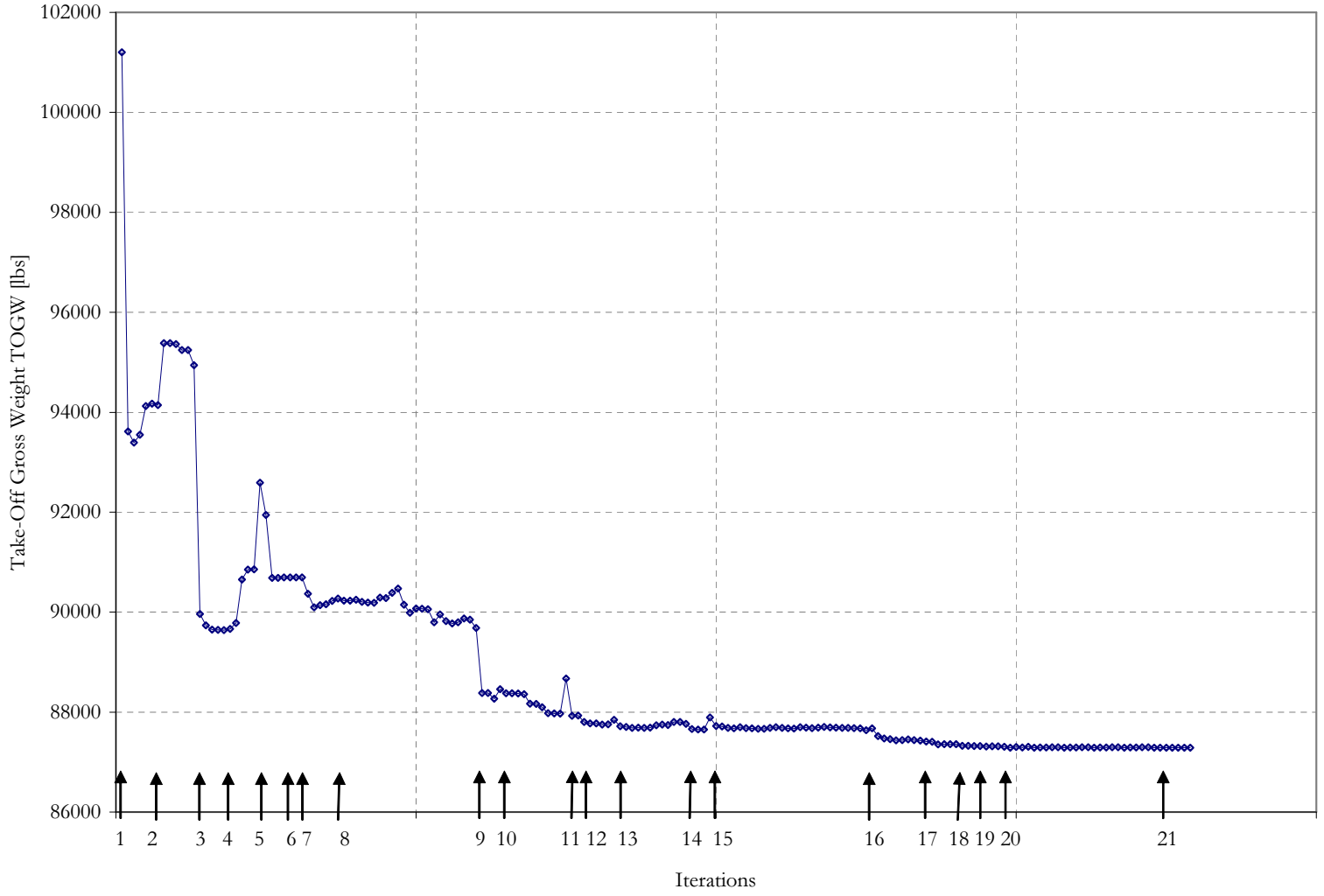

Figure 121: Second Vehicle - Take-off Gross Weight

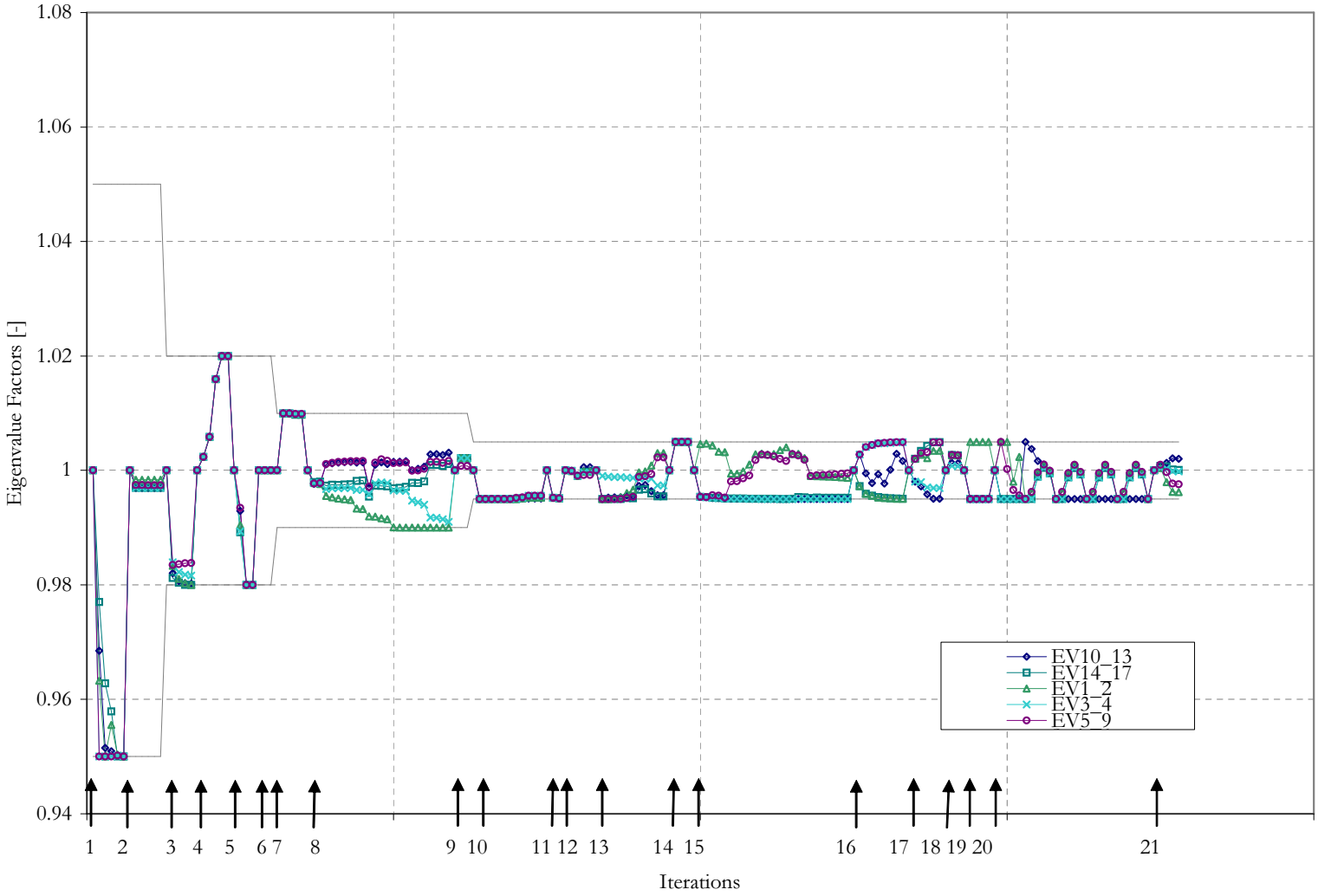

Figure 122: Second Vehicle - Eigenvalue Pooling Factors

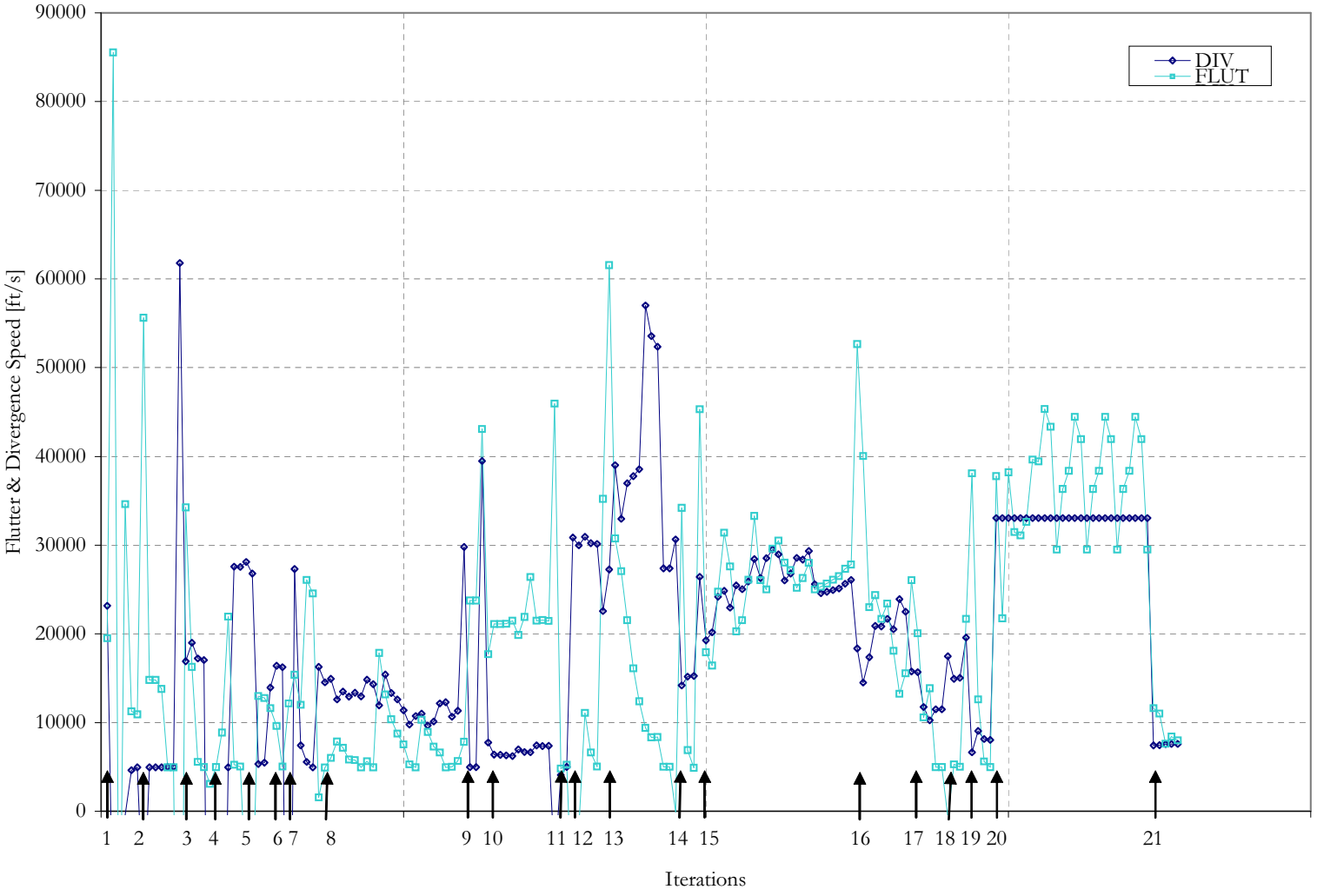

Figure 123: Second Vehicle - Flutter and Divergence Speed

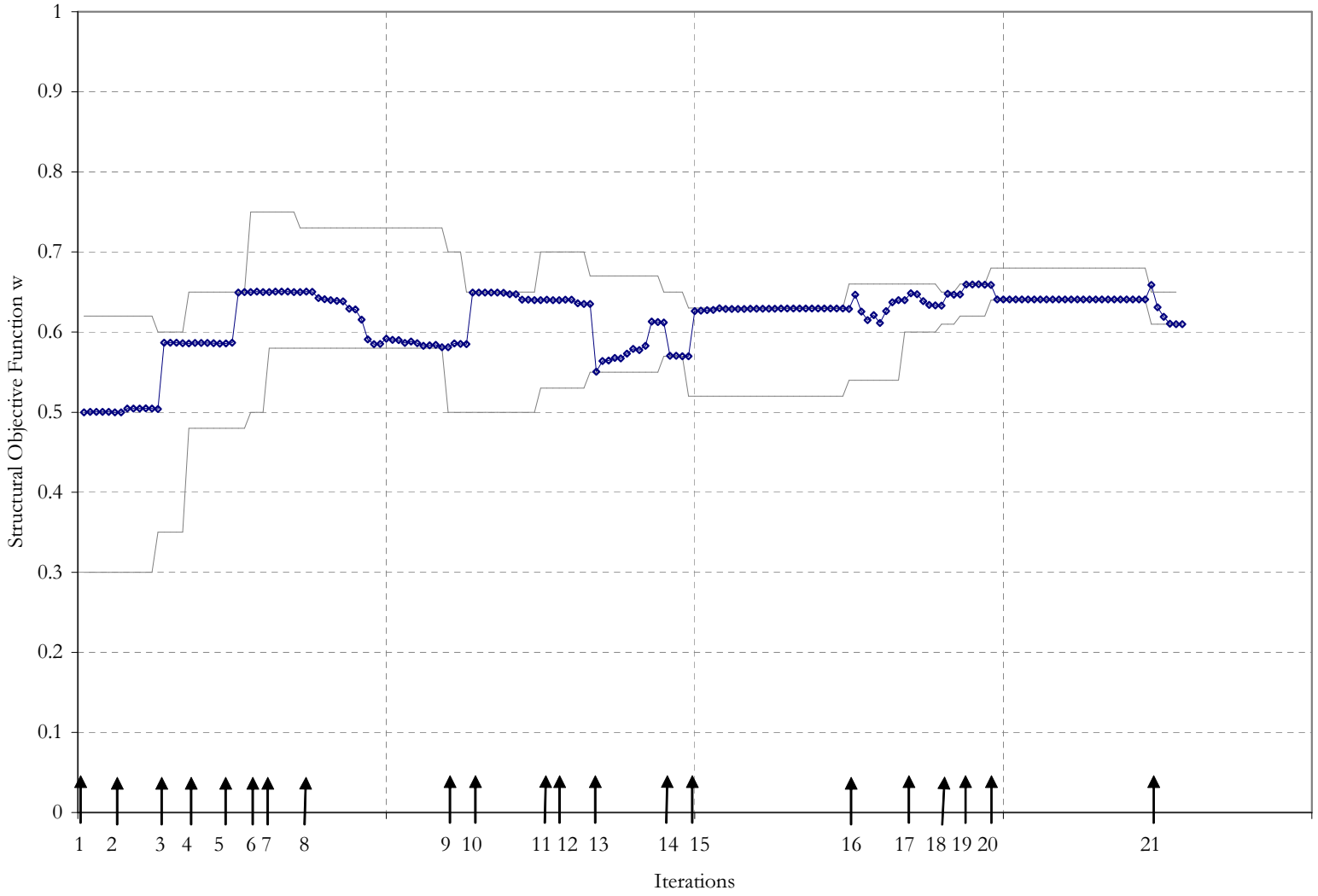

Figure 124: Second Vehicle - Structural Objective Weight w

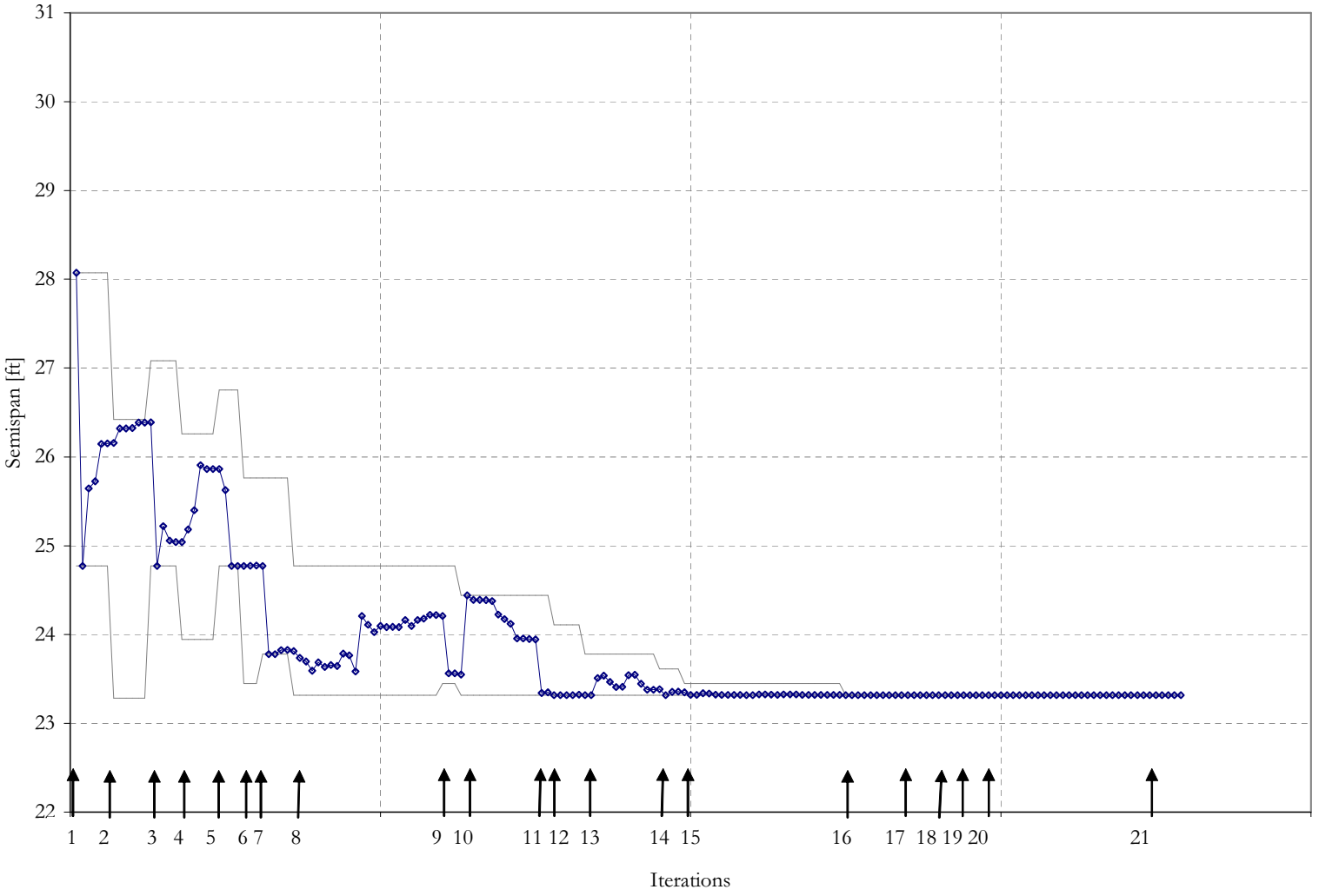

Figure 125: Second Vehicle - Geometric Variables, Semi-span

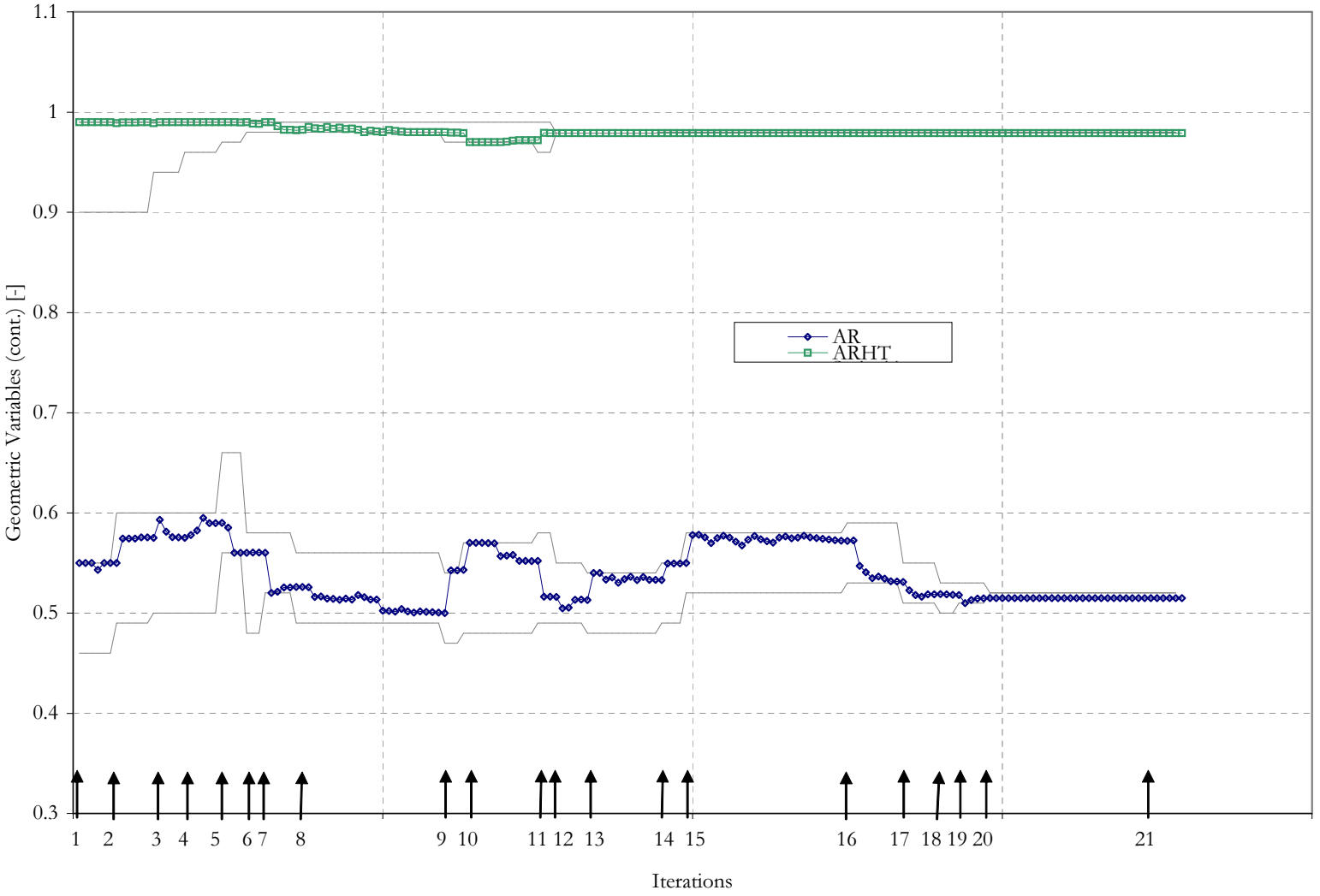

Figure 126: Second Vehicle - Geometric Variables, Aspect Ratio

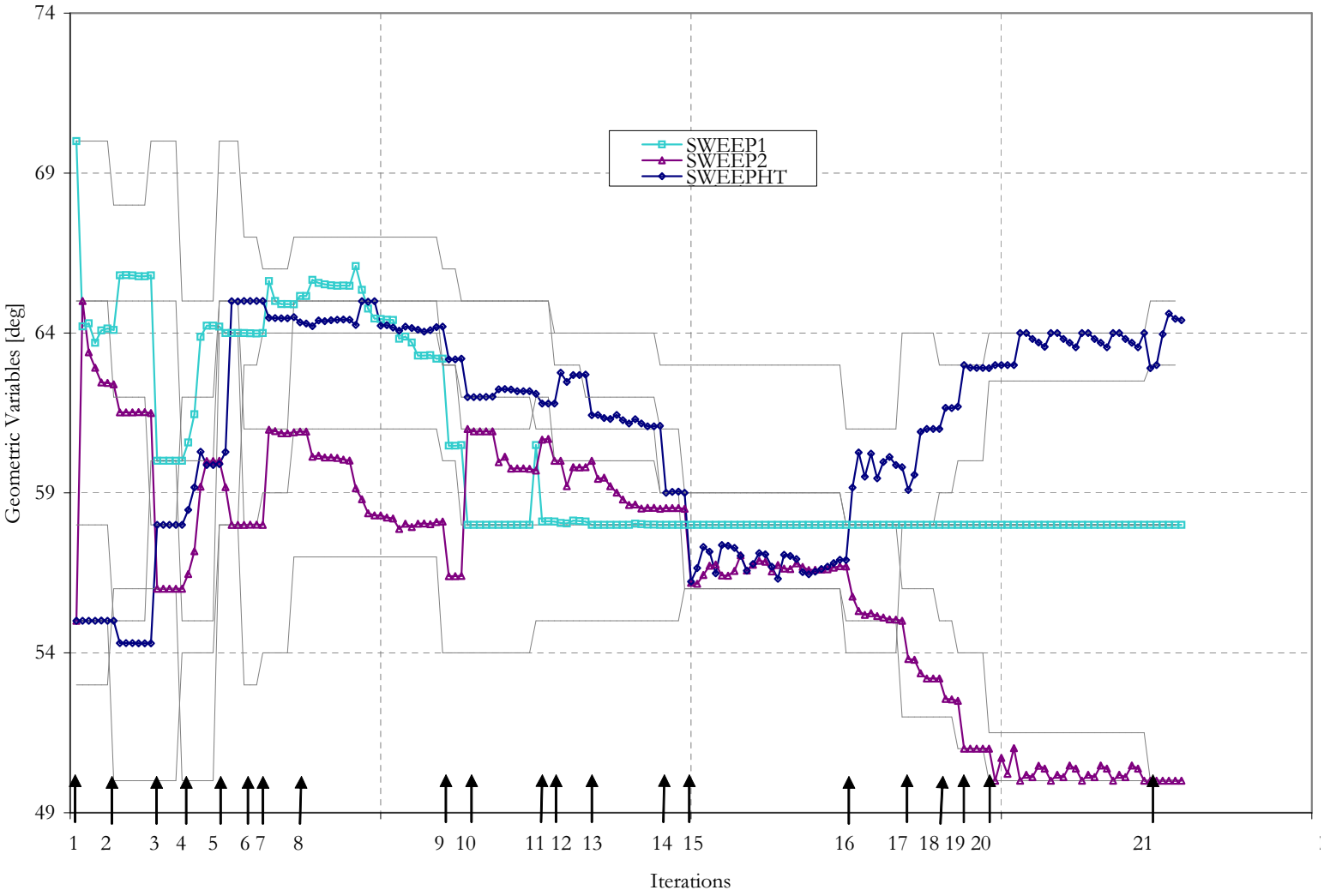

Figure 127: Second Vehicle - Geometric Variables, Sweep

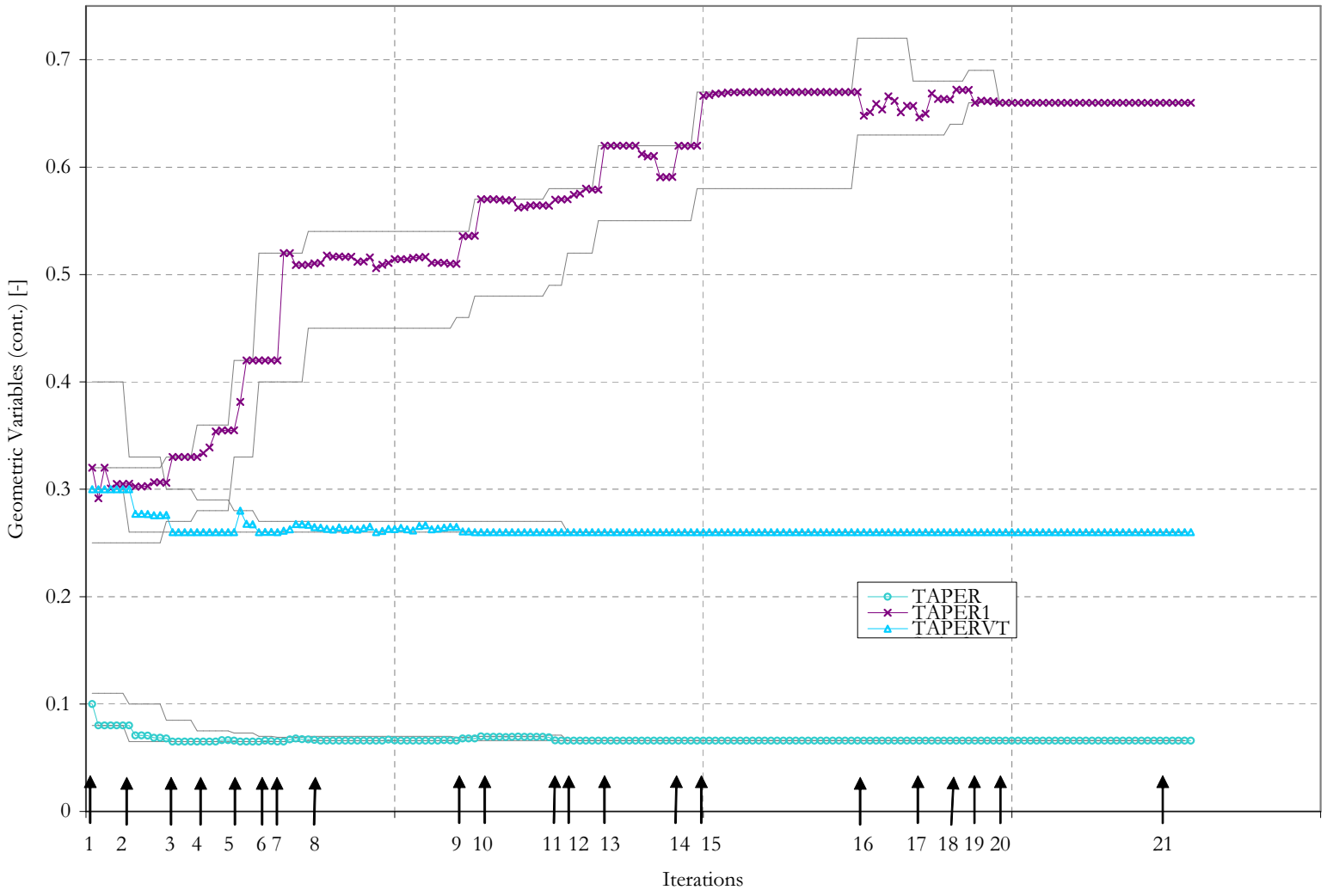

Figure 128: Second Vehicle - Geometric Variables, Taper

D.3 Third Optimization Run

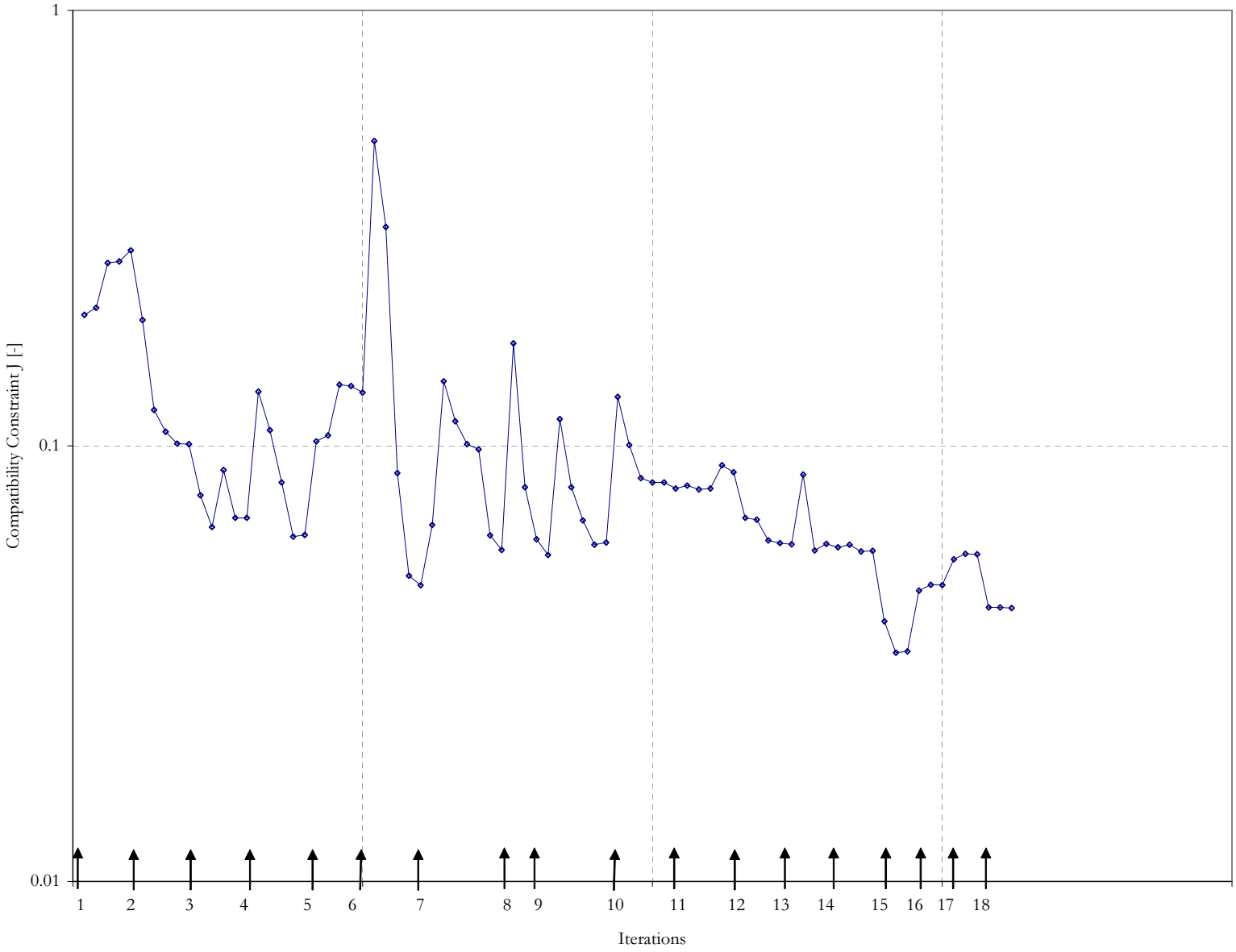

Figure 129: Third Vehicle - Compatibility Constraint J

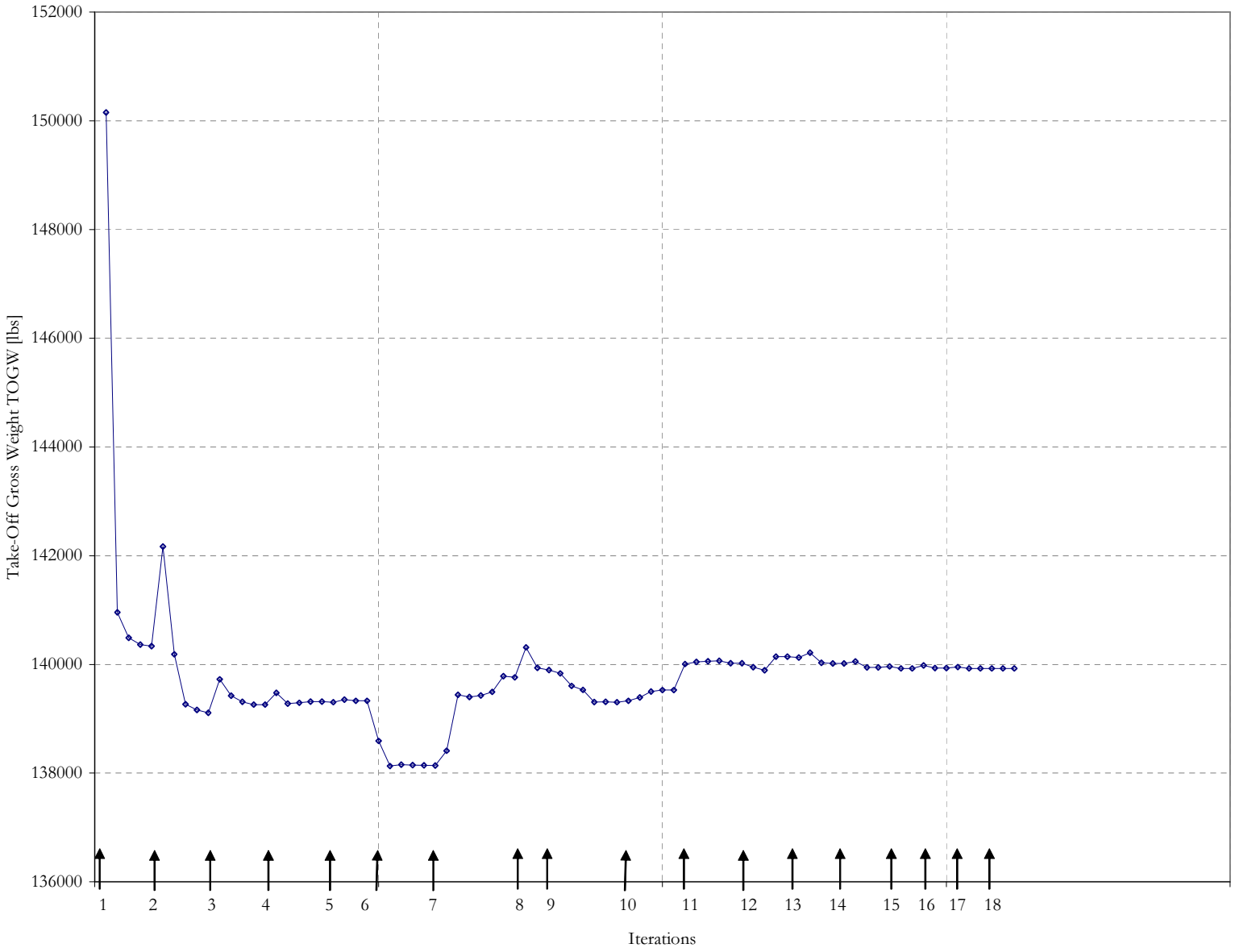

Figure 130: Third Vehicle - Take-off Gross Weight

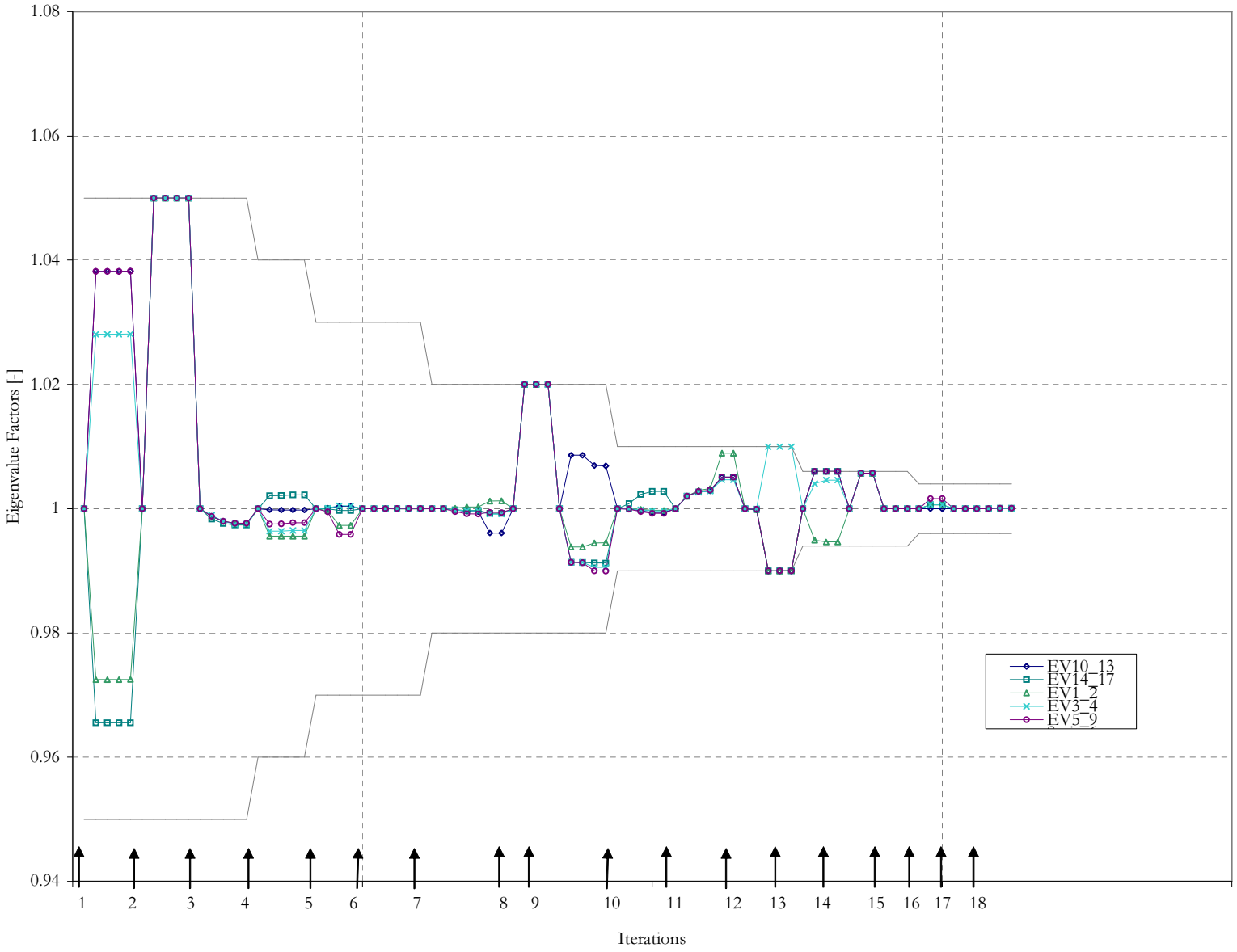

Figure 131: Third Vehicle - Eigenvalue Pooling Factors

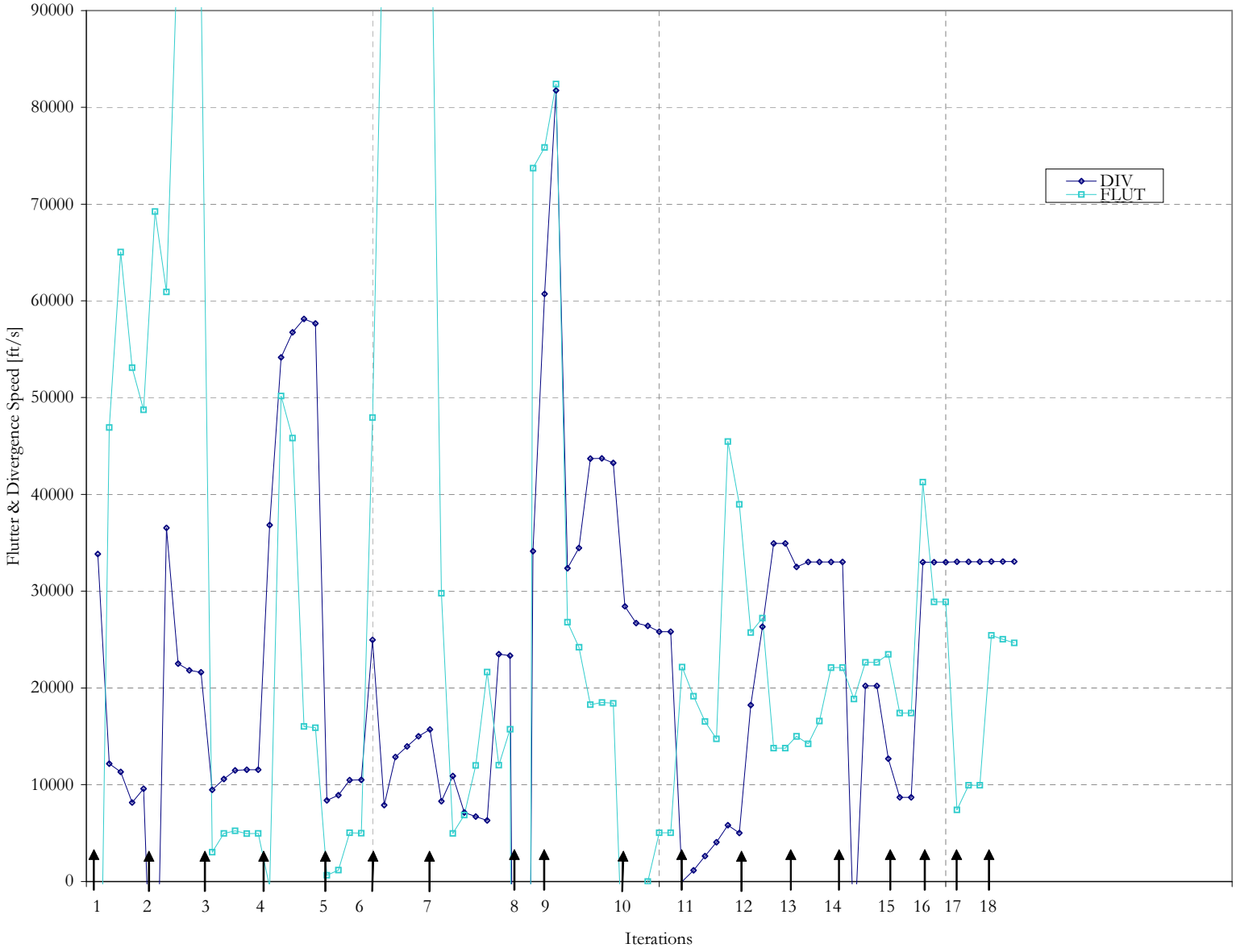

Figure 132: Third Vehicle - Flutter and Divergence Speed

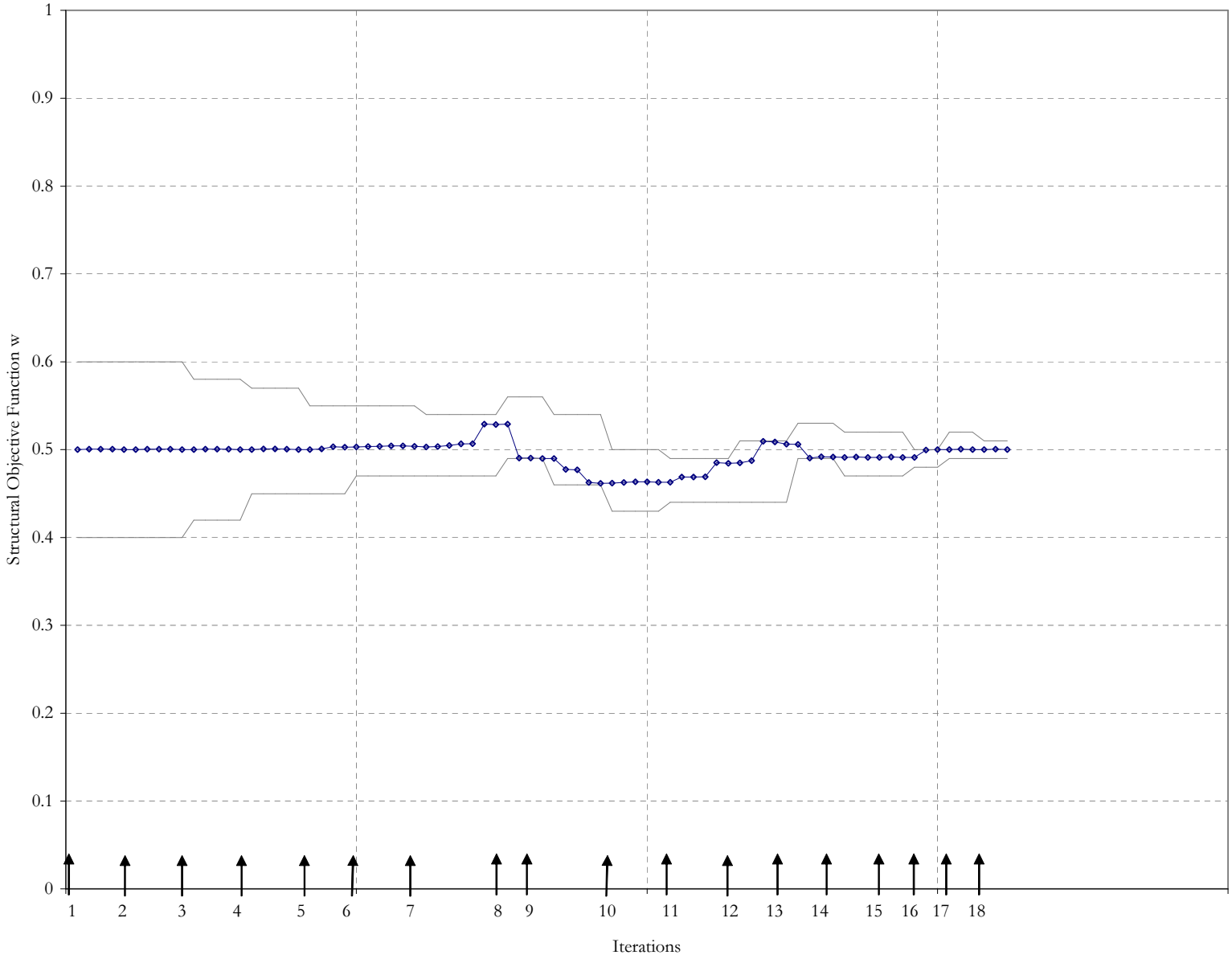

Figure 133: Third Vehicle - Structural Objective Weight w

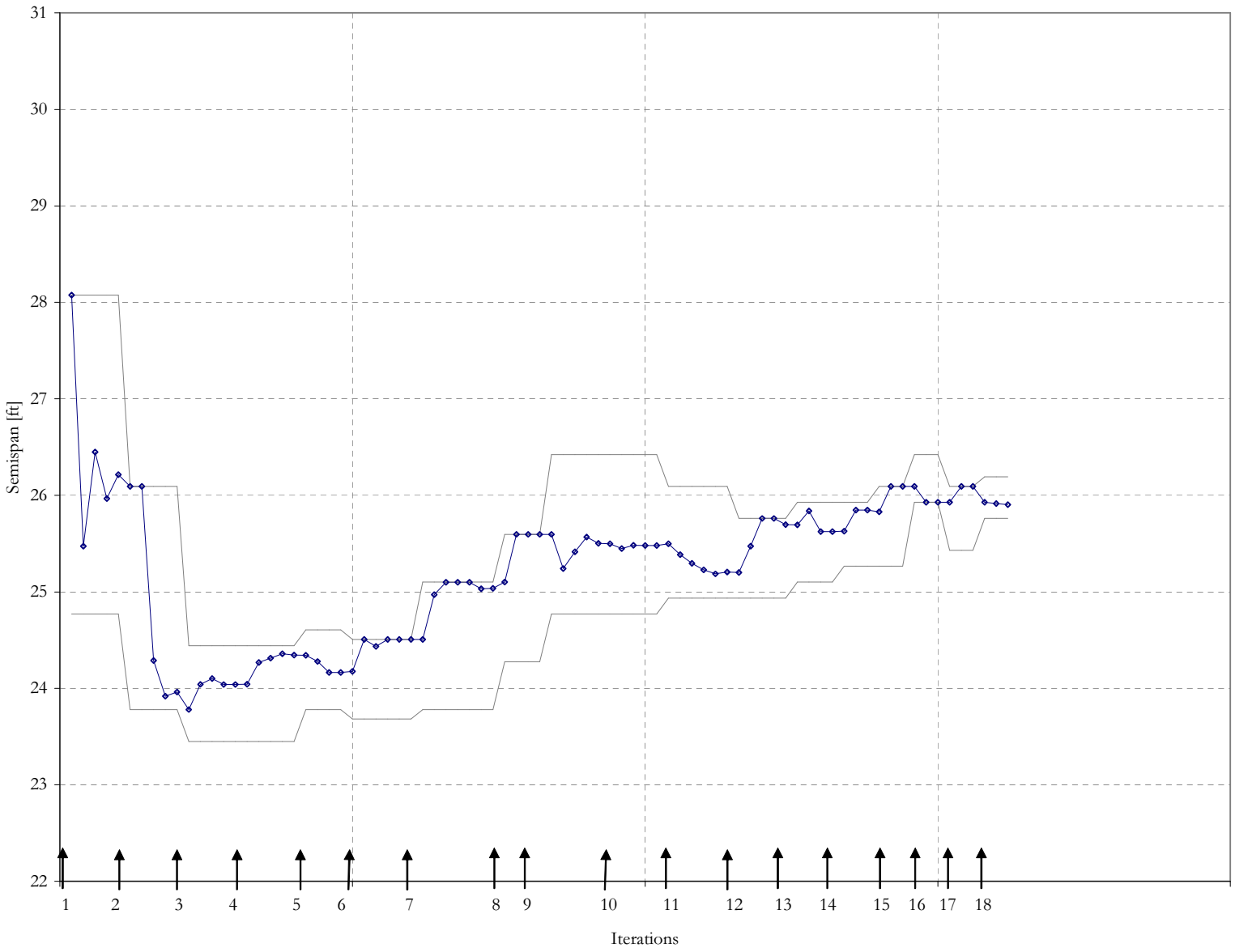

Figure 134: Third Vehicle - Geometric Variables, Semi-span

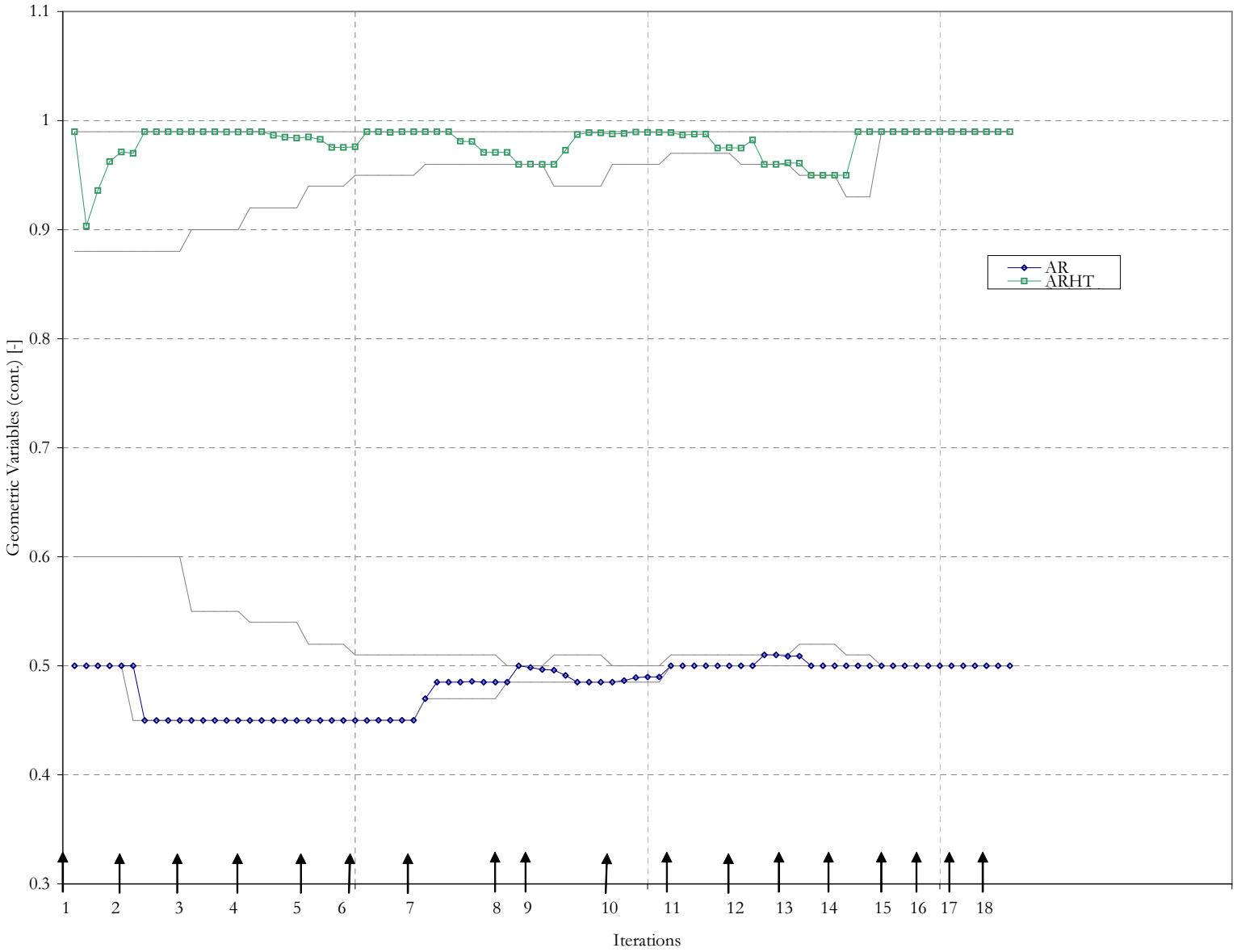

Figure 135: Third Vehicle - Geometric Variables, Aspect Ratio

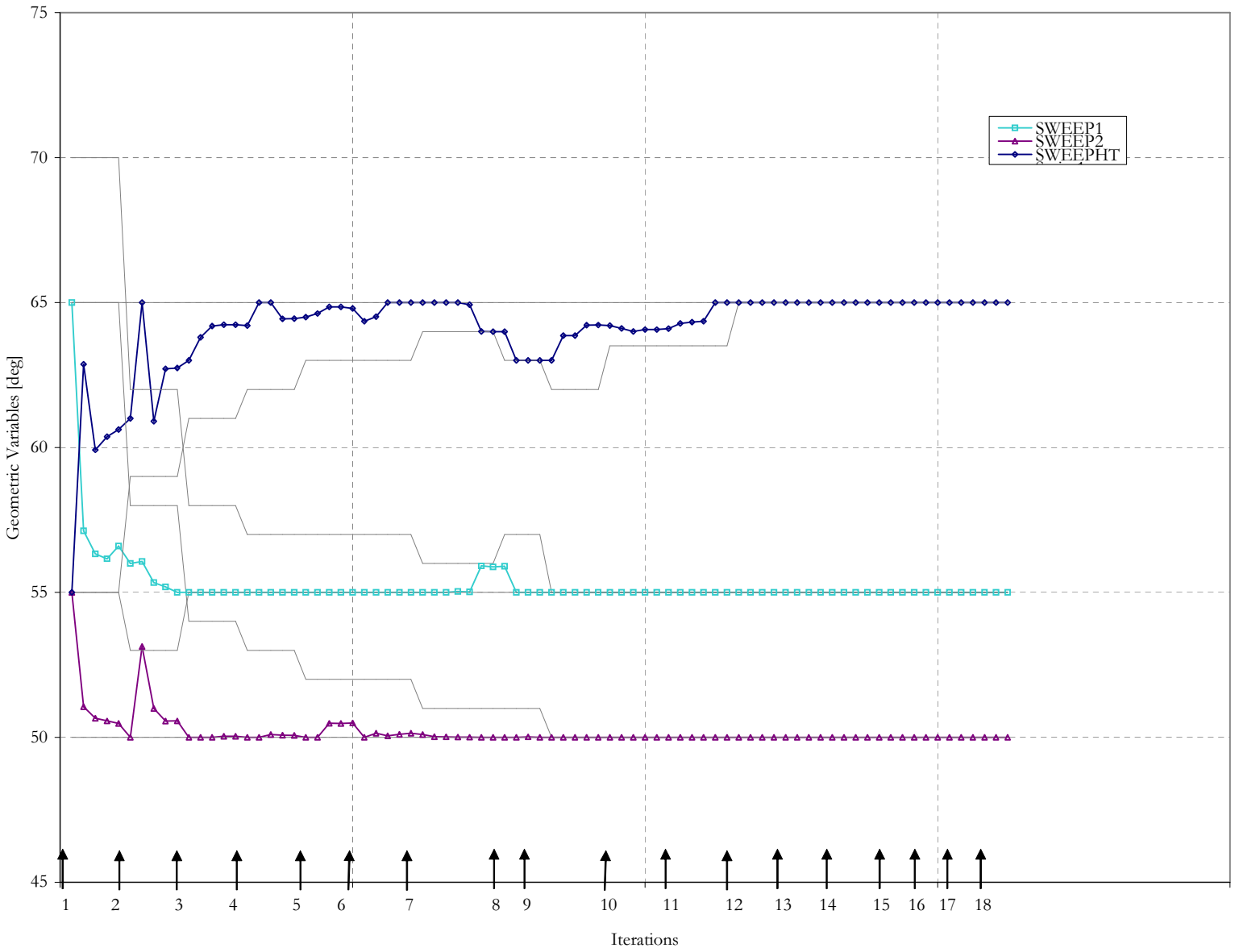

Figure 136: Third Vehicle - Geometric Variables, Sweep

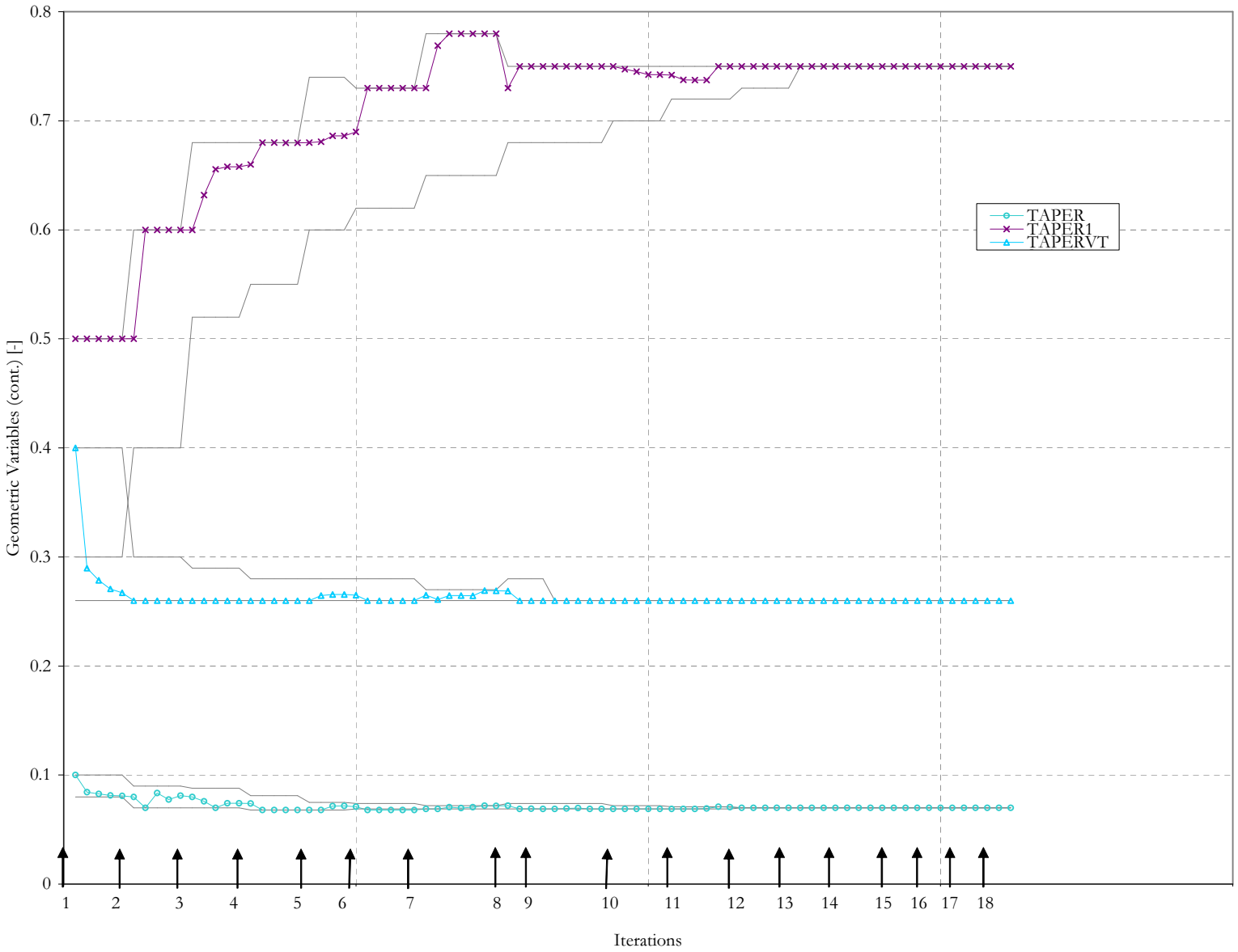

Figure 137: Third Vehicle - Geometric Variables, Taper

D.4 Fourth Optimization Run

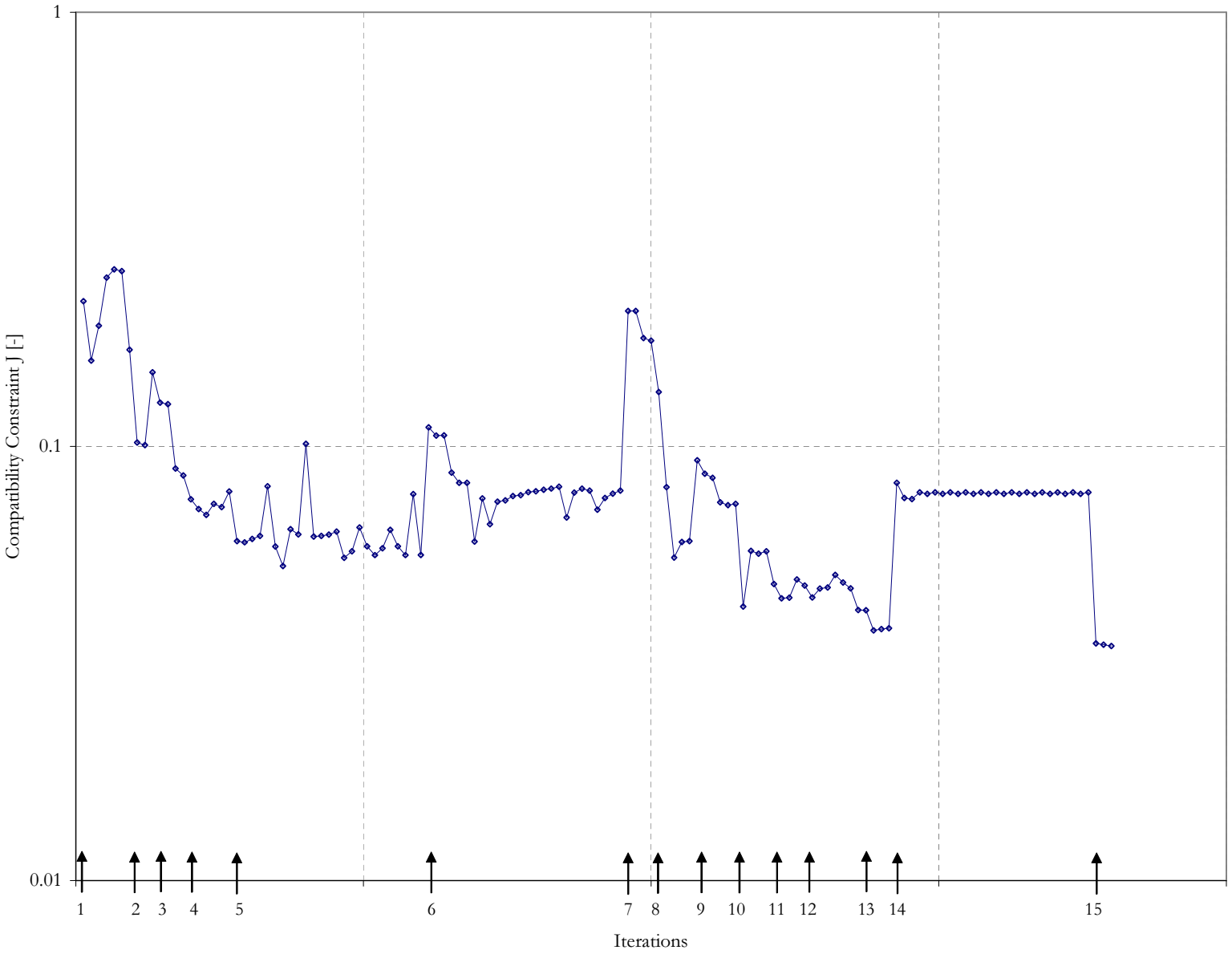

Figure 138: Fourth Vehicle - Compatibility Constraint J

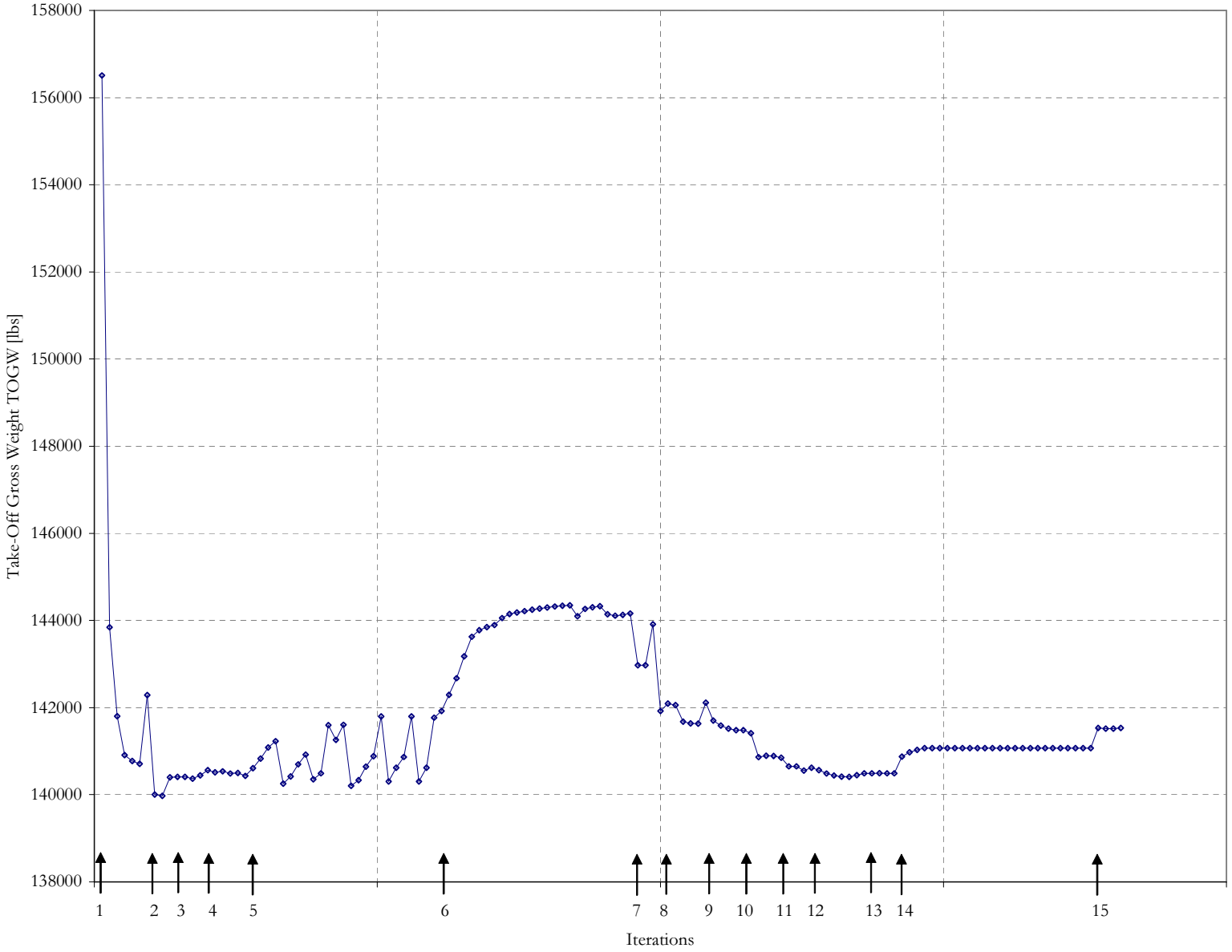

Figure 139: Fourth Vehicle - Take-off Gross Weight

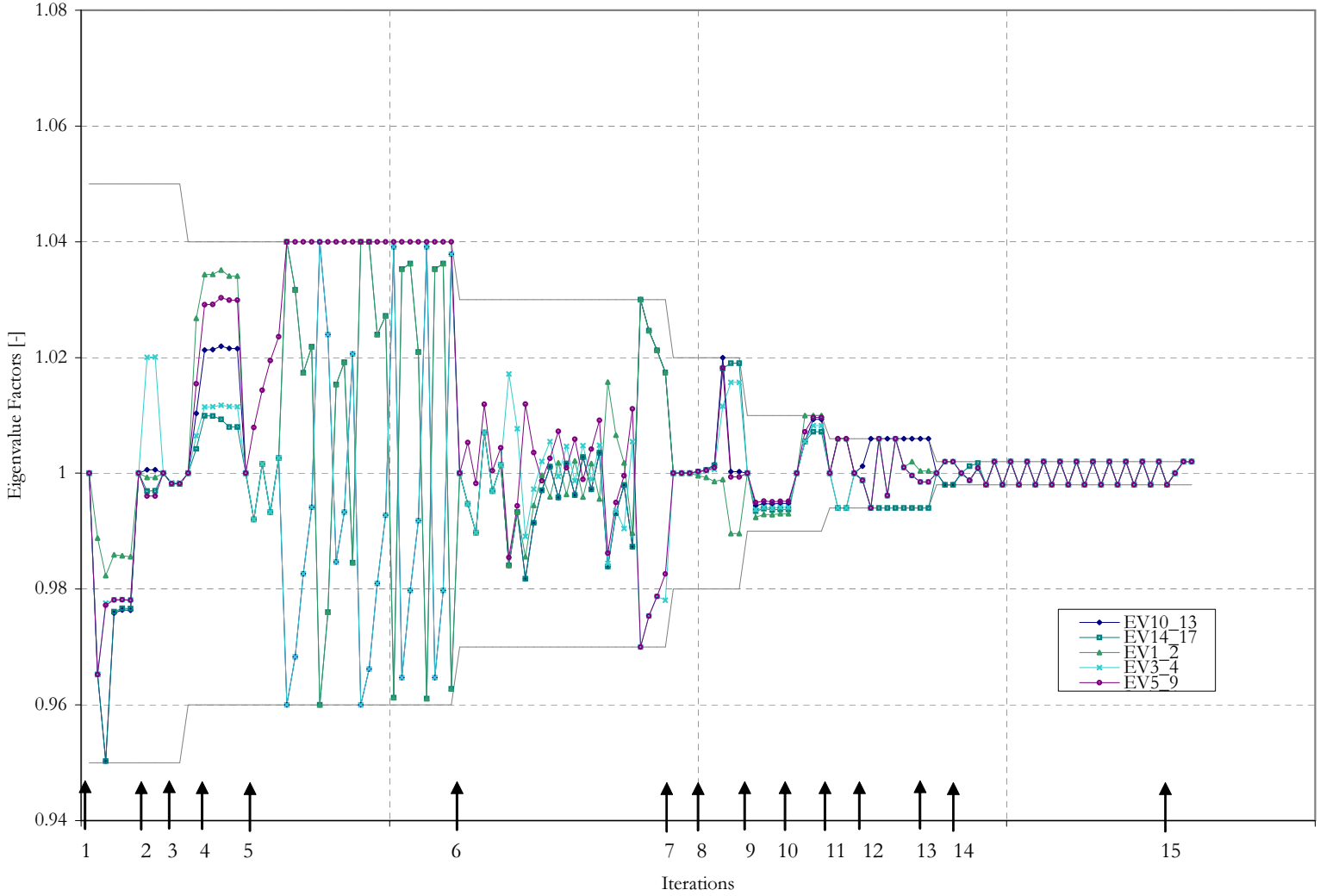

Figure 140: Fourth Vehicle - Eigenvalue Pooling Factors

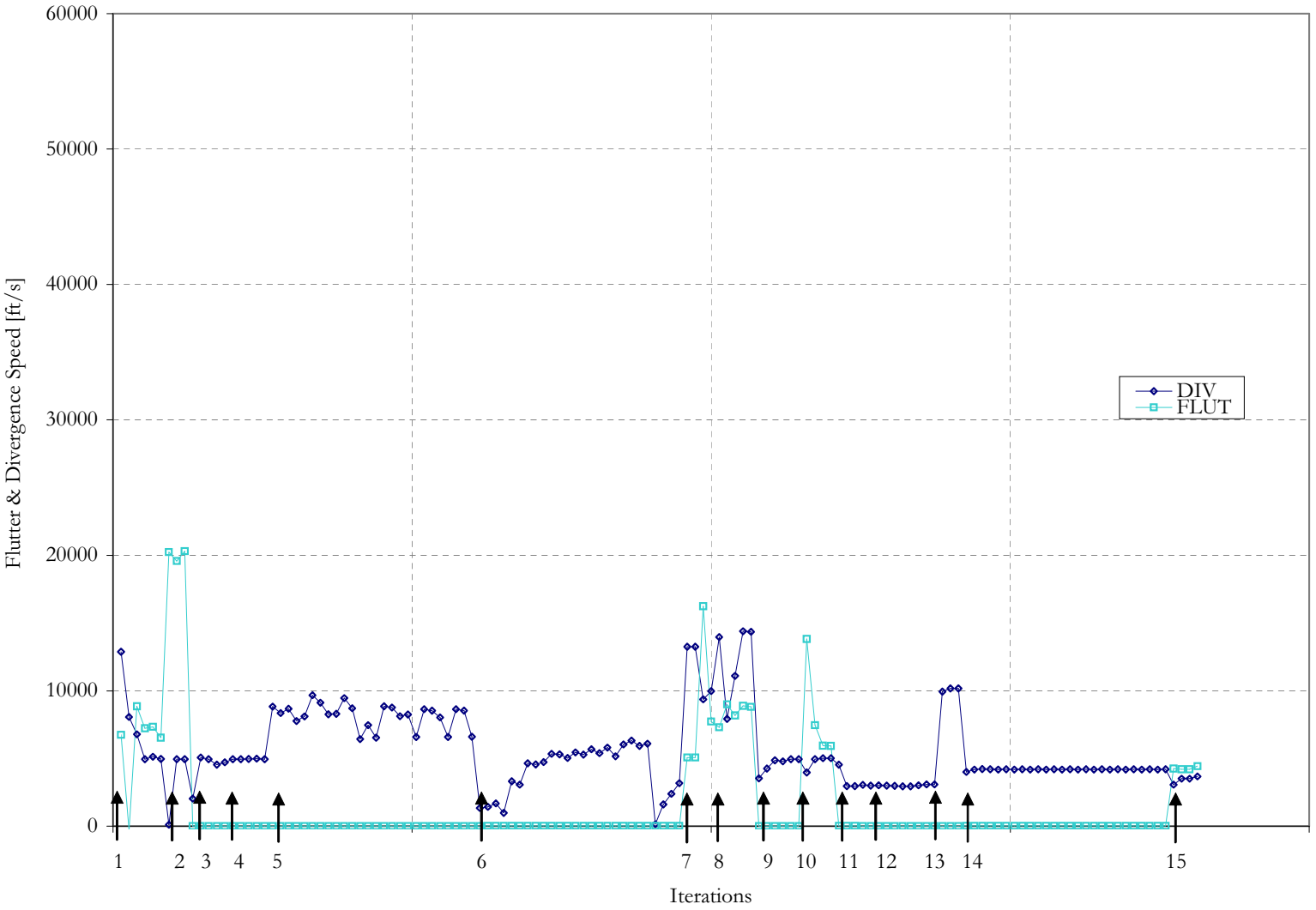

Figure 141: Fourth Vehicle - Flutter and Divergence Speed

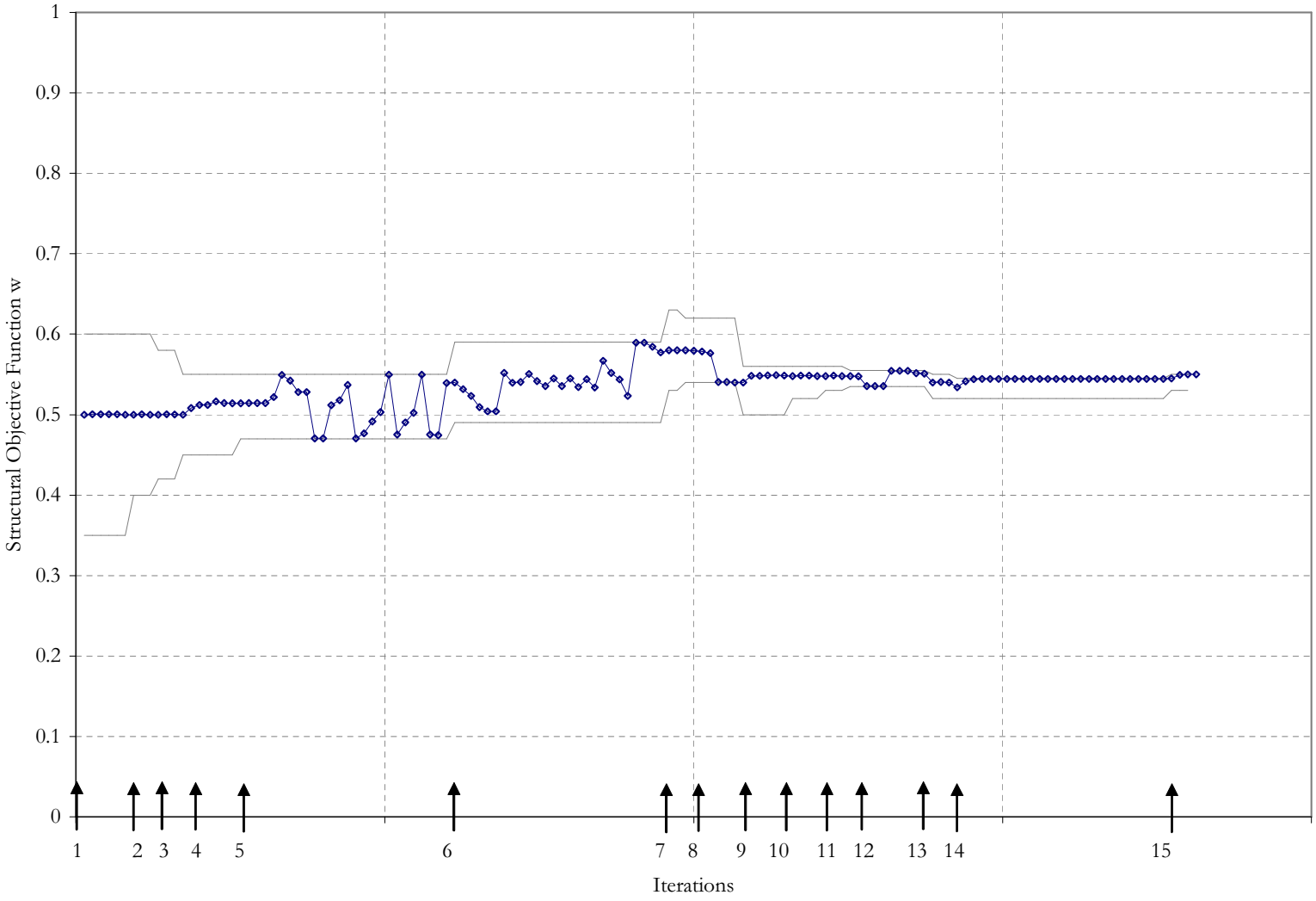

Figure 142: Fourth Vehicle - Structural Objective Weight w

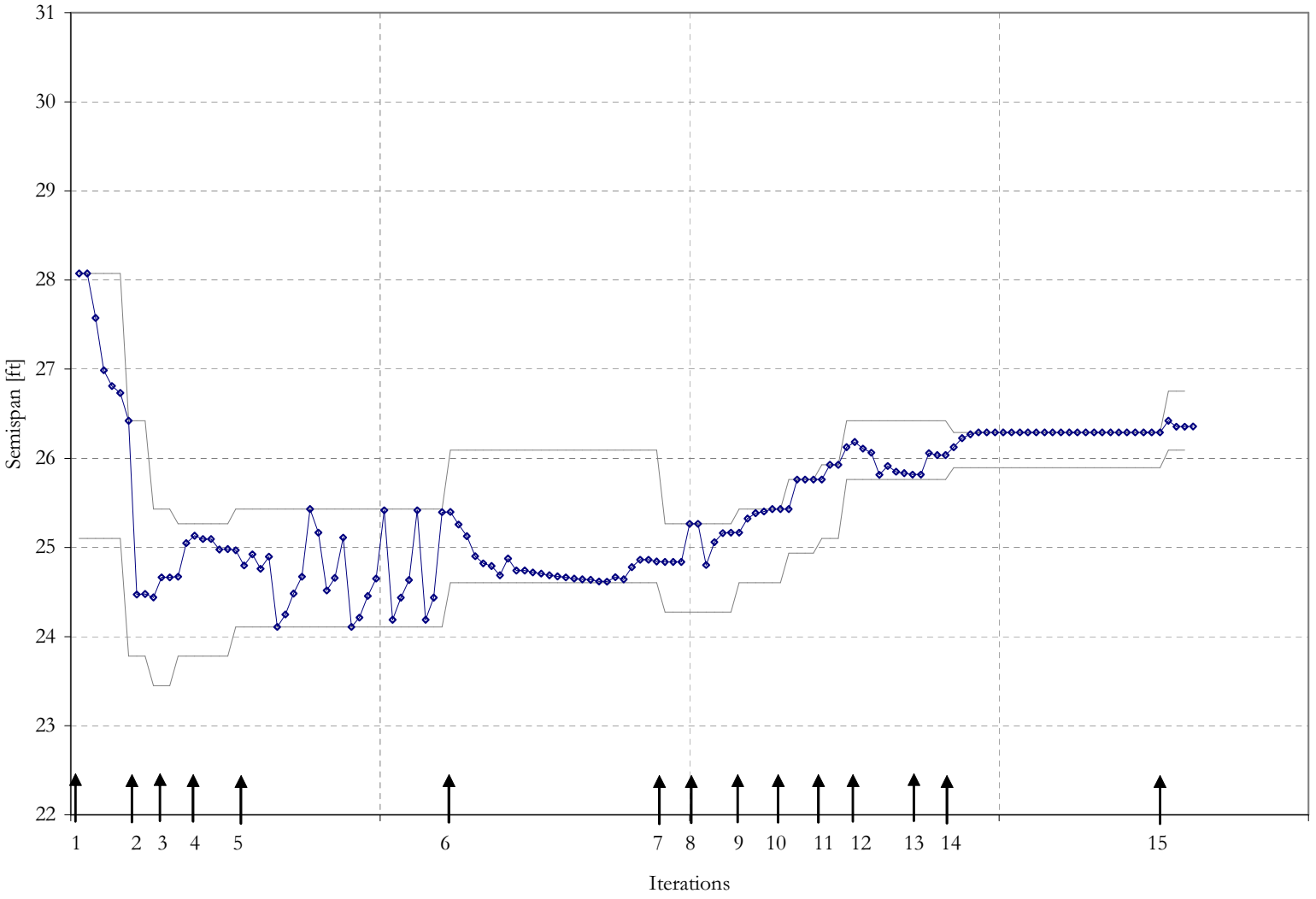

Figure 143: Fourth Vehicle - Geometric Variables, Semi-span

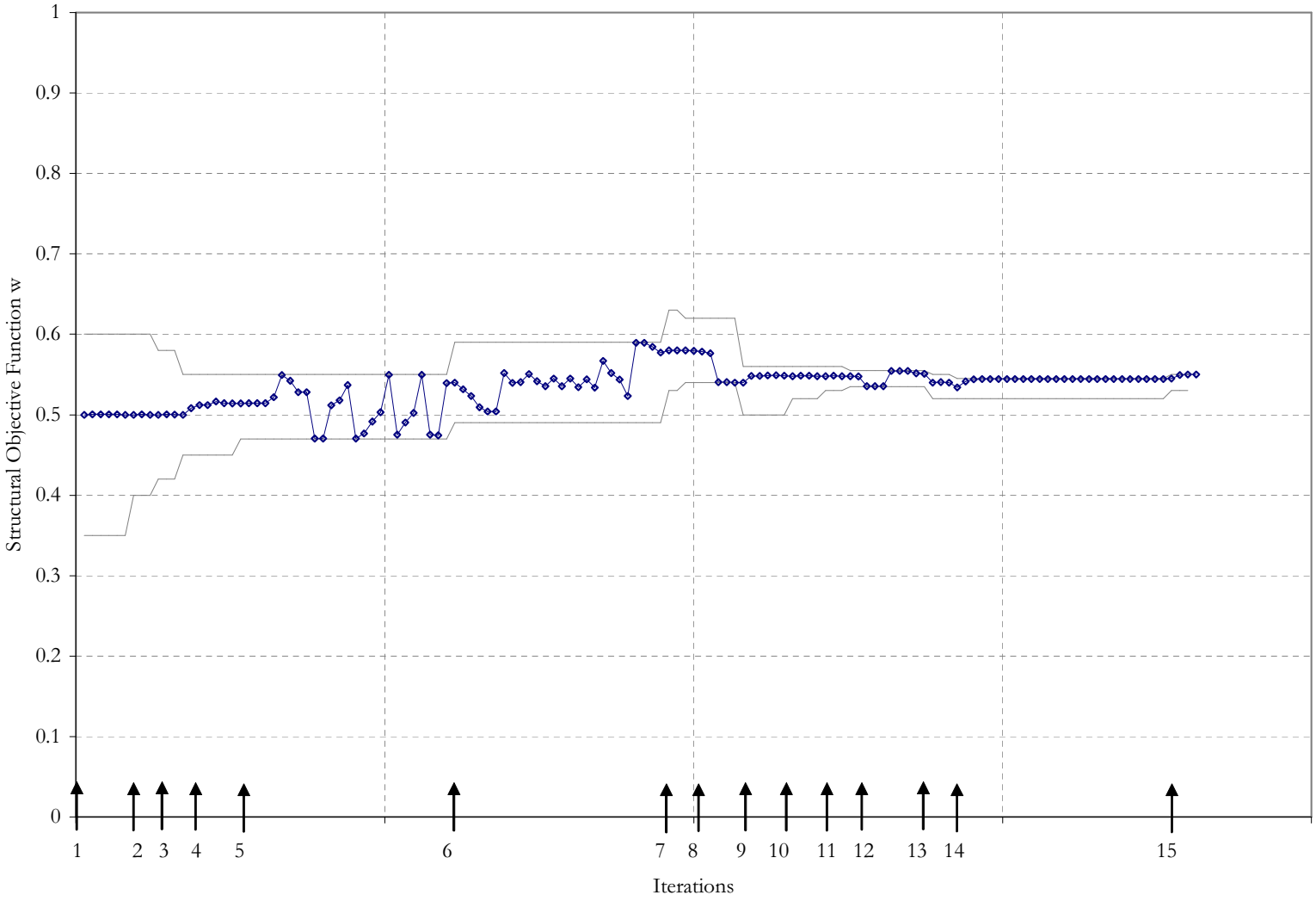

Figure 144: Fourth Vehicle - Geometric Variables, Aspect Ratio

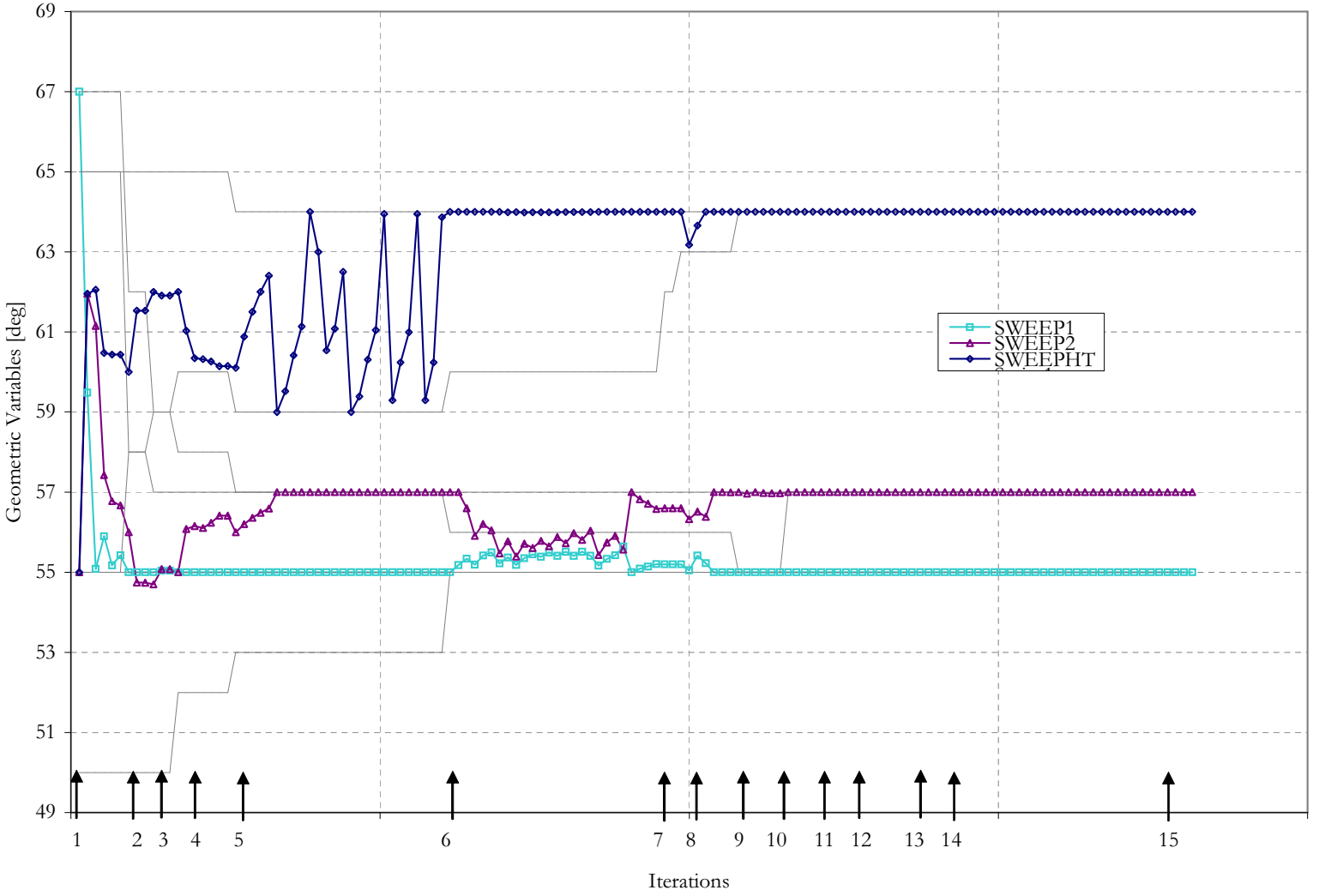

Figure 145: Fourth Vehicle - Geometric Variables, Sweep

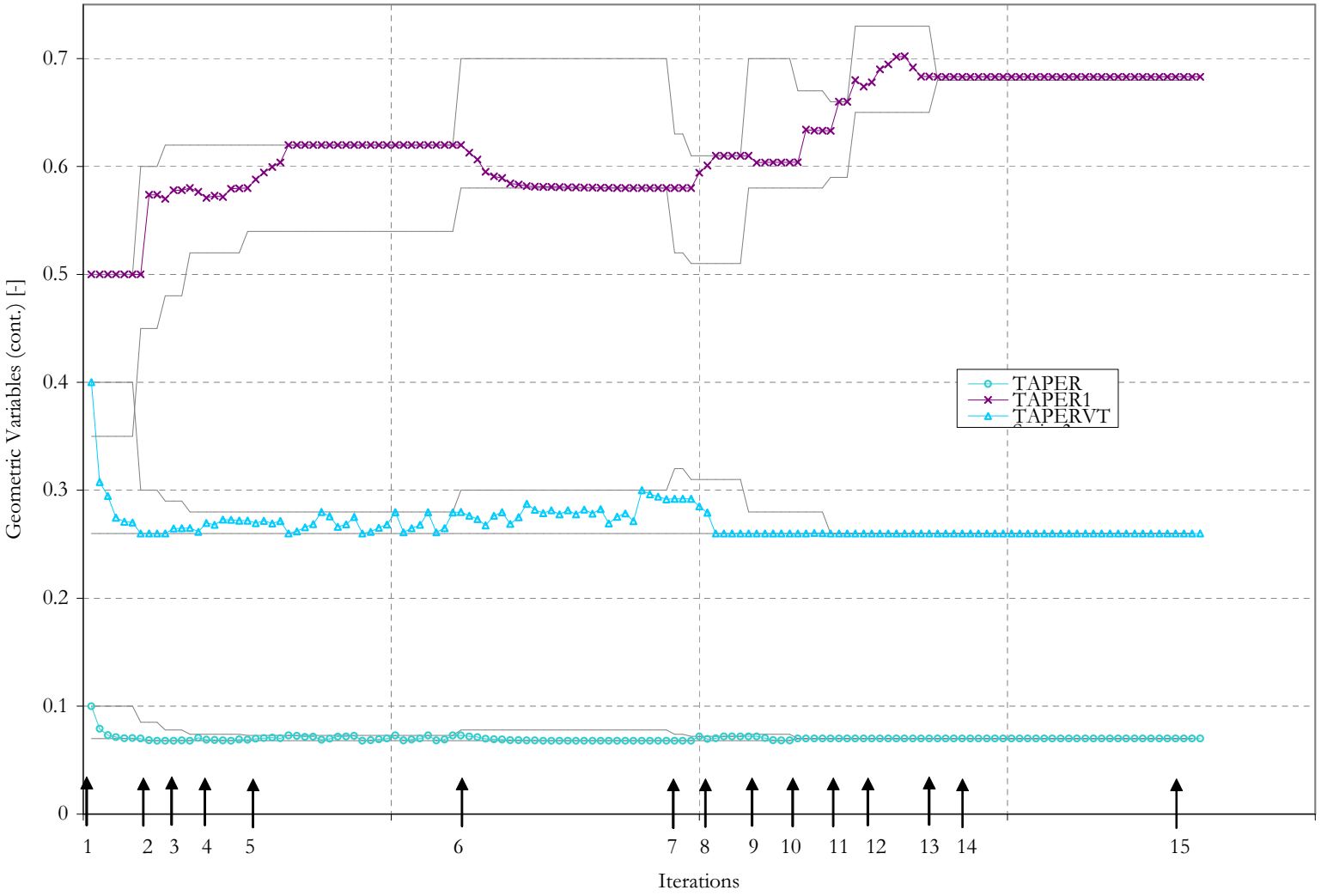

Figure 146: Fourth Vehicle - Geometric Variables, Taper

D.5 Subsonic Optimization Run

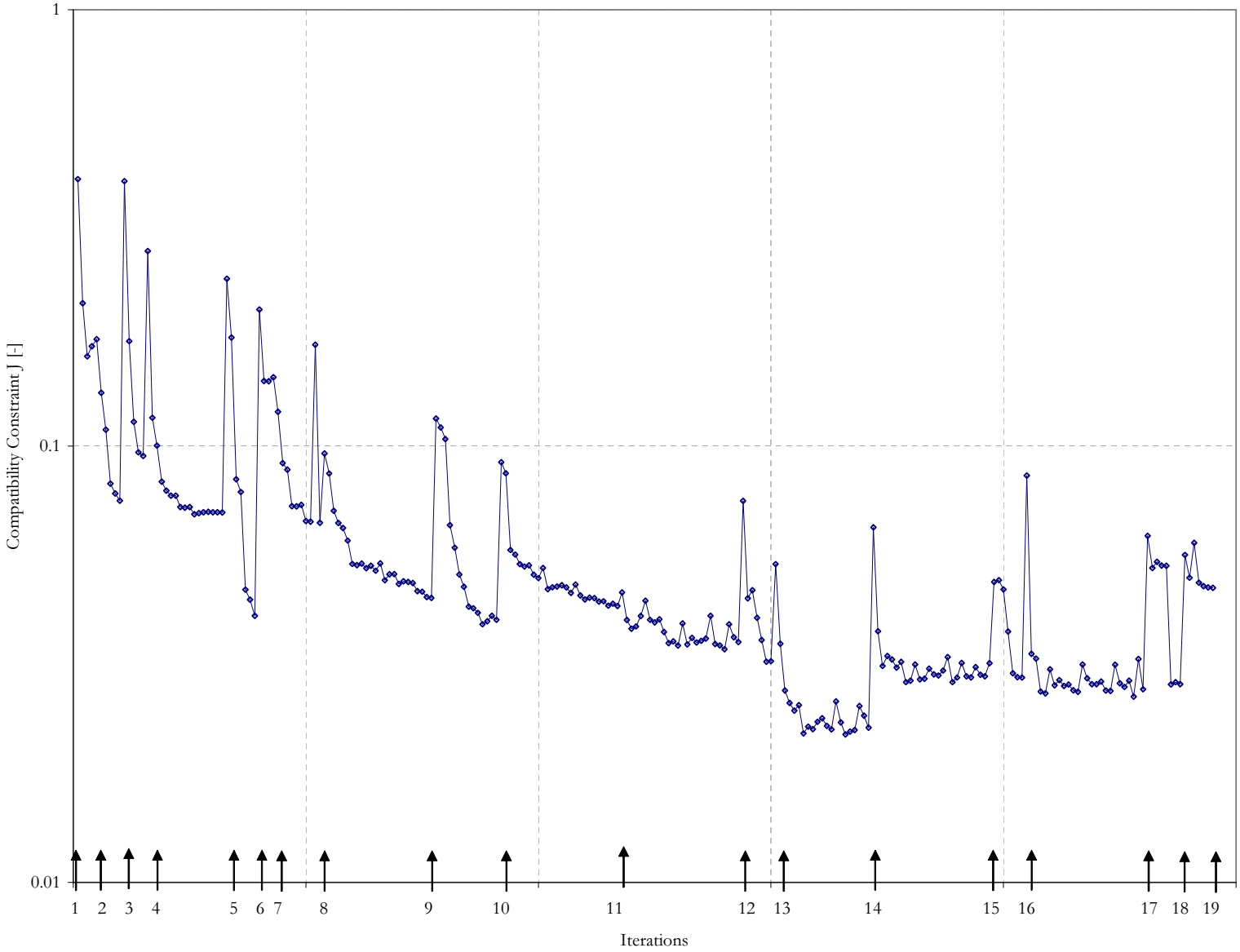

Figure 147: Subsonic Vehicle - Compatibility Constraint J

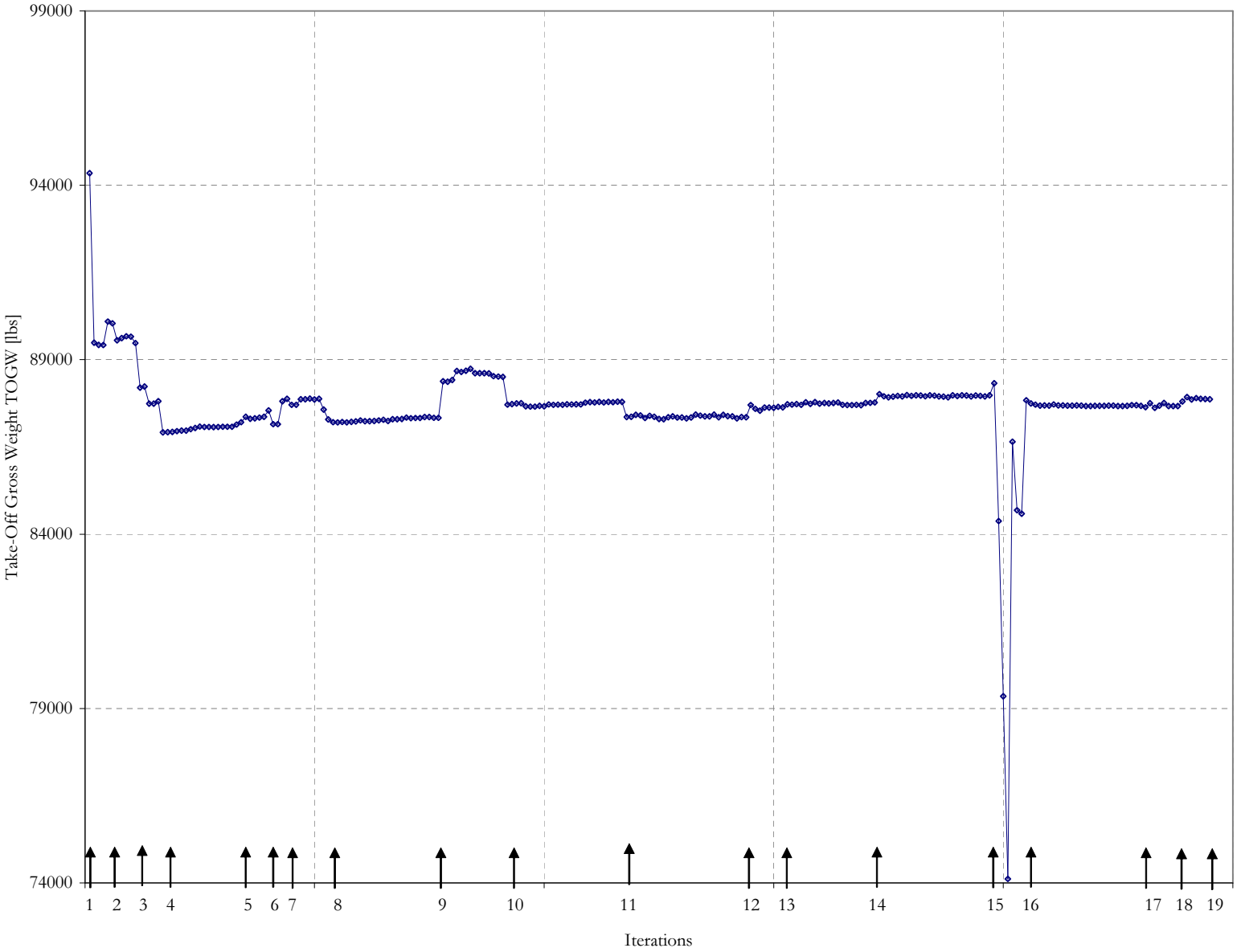

Figure 148: Subsonic Vehicle - Take-off Gross Weight

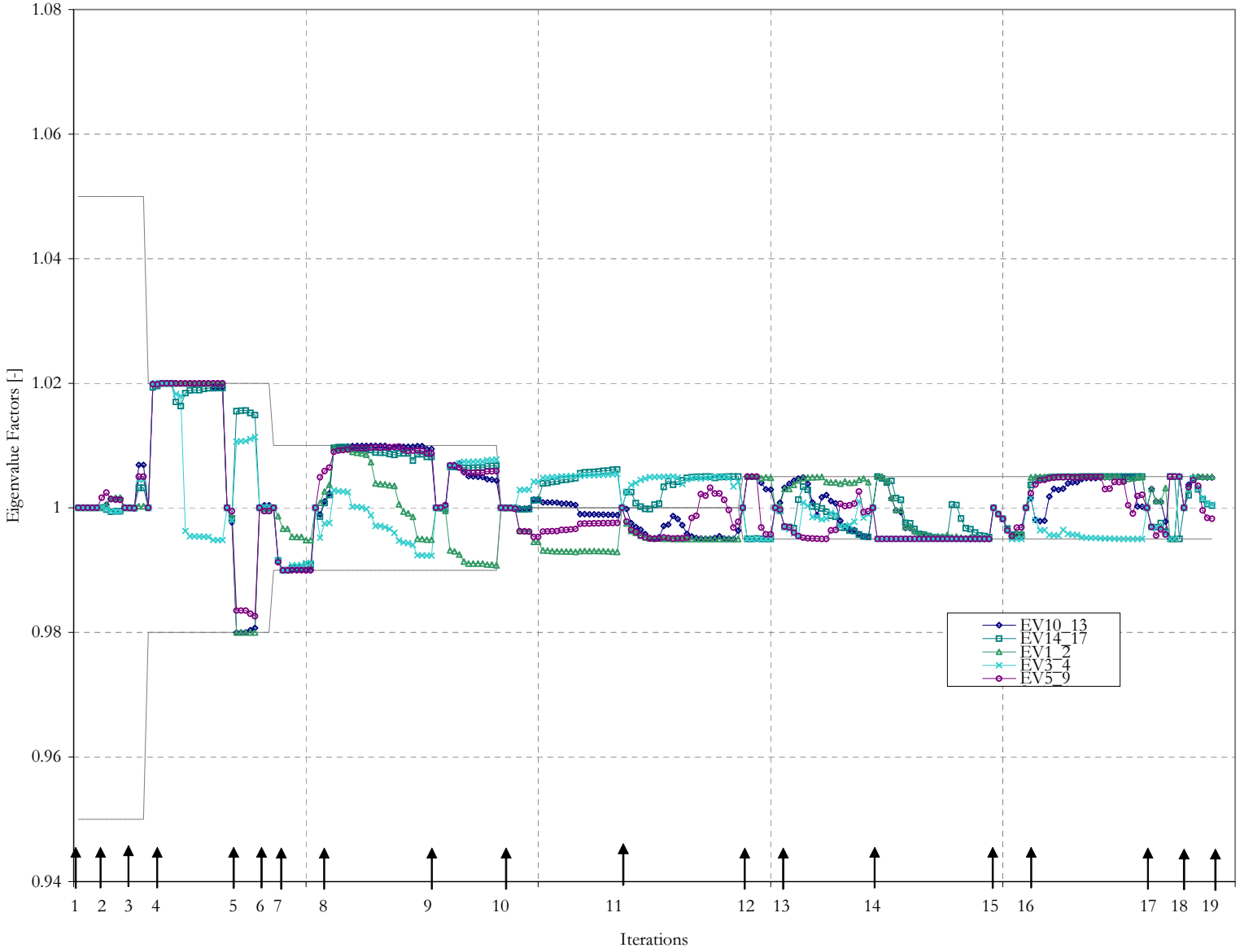

Figure 149: Subsonic Vehicle - Eigenvalue Pooling Factors

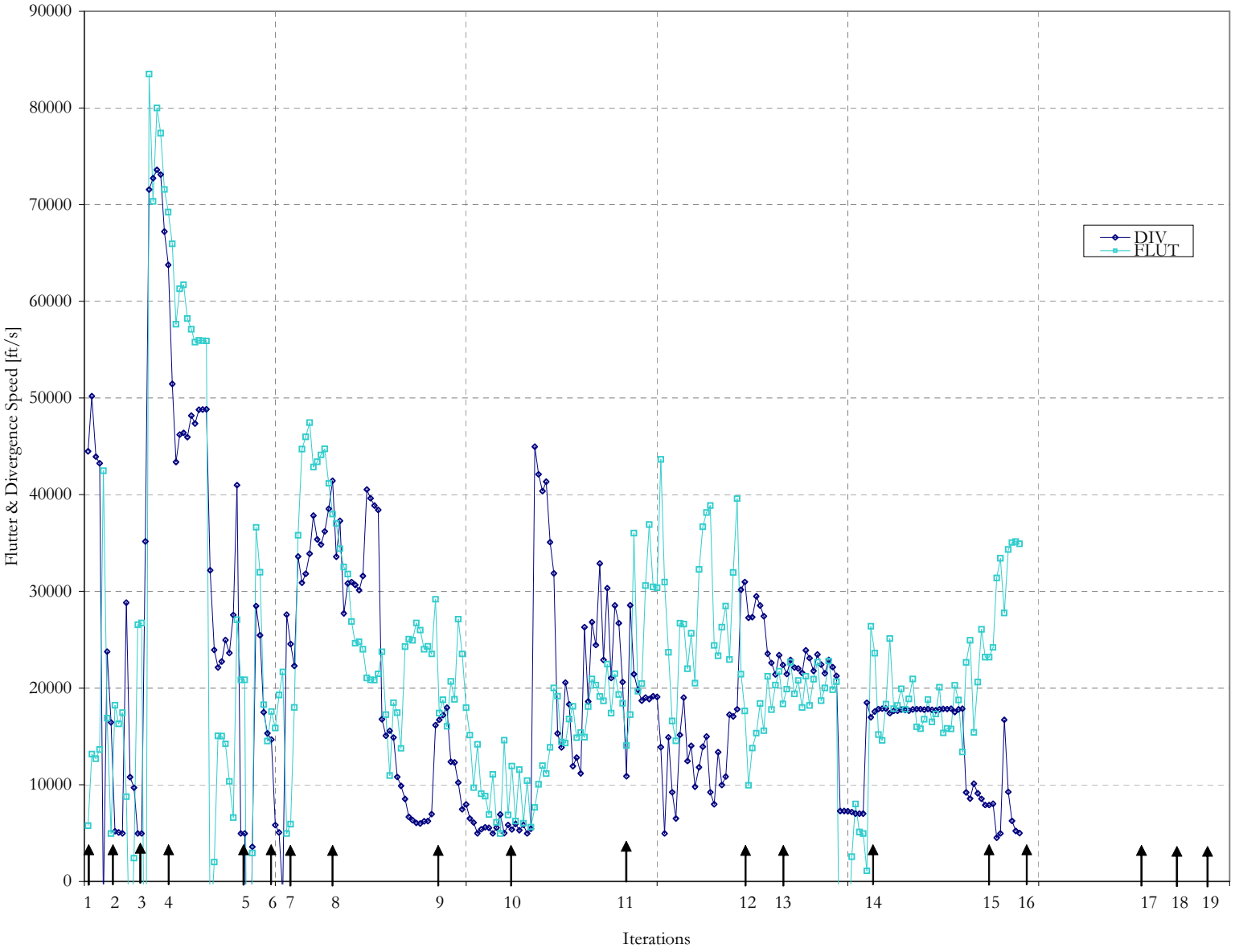

Figure 150: Subsonic Vehicle - Flutter and Divergence Speed

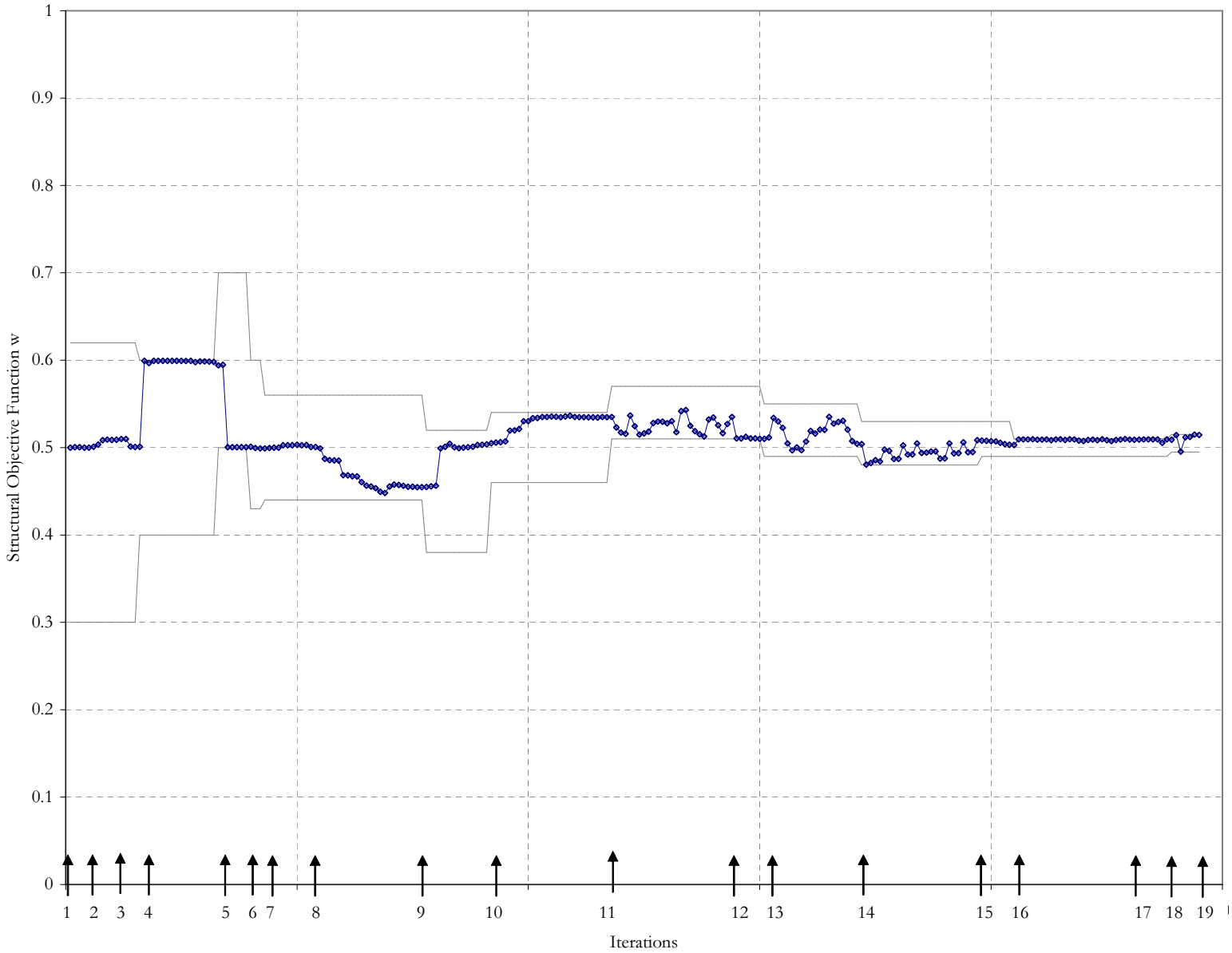

Figure 151: Subsonic Vehicle - Structural Objective Weight w

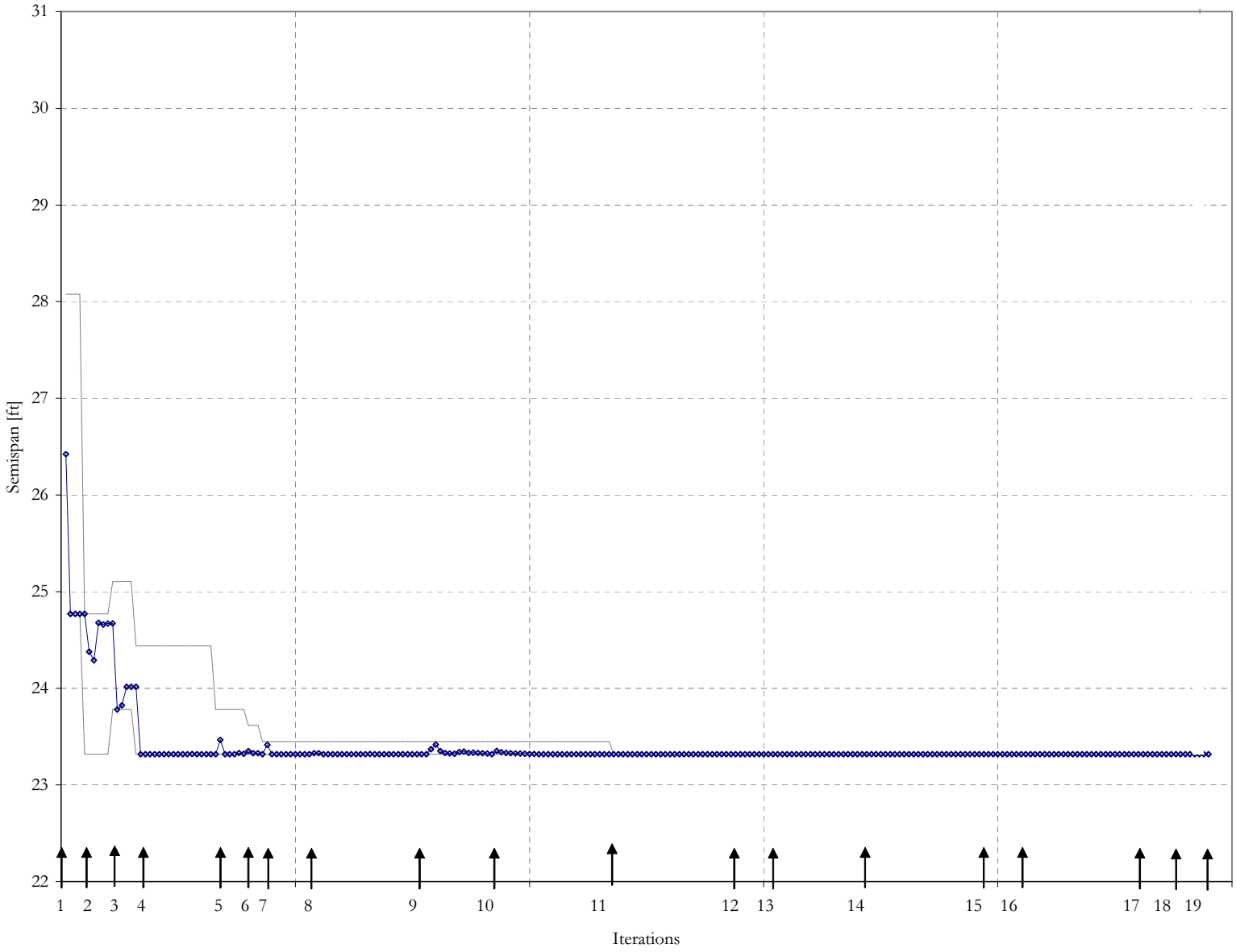

Figure 152: Subsonic Vehicle - Geometric Variables, Semi-span

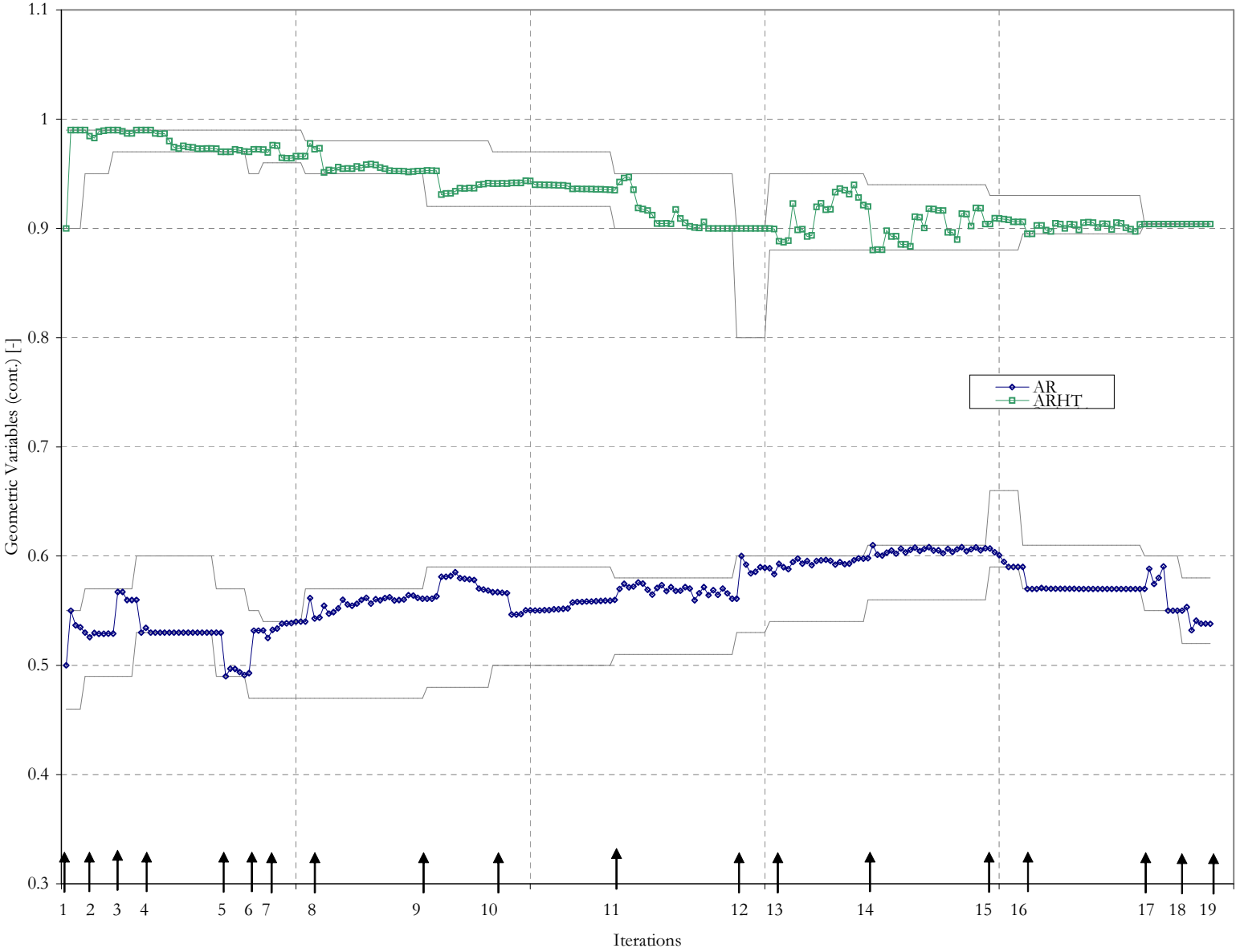

Figure 153: Subsonic Vehicle - Geometric Variables, Aspect Ratio

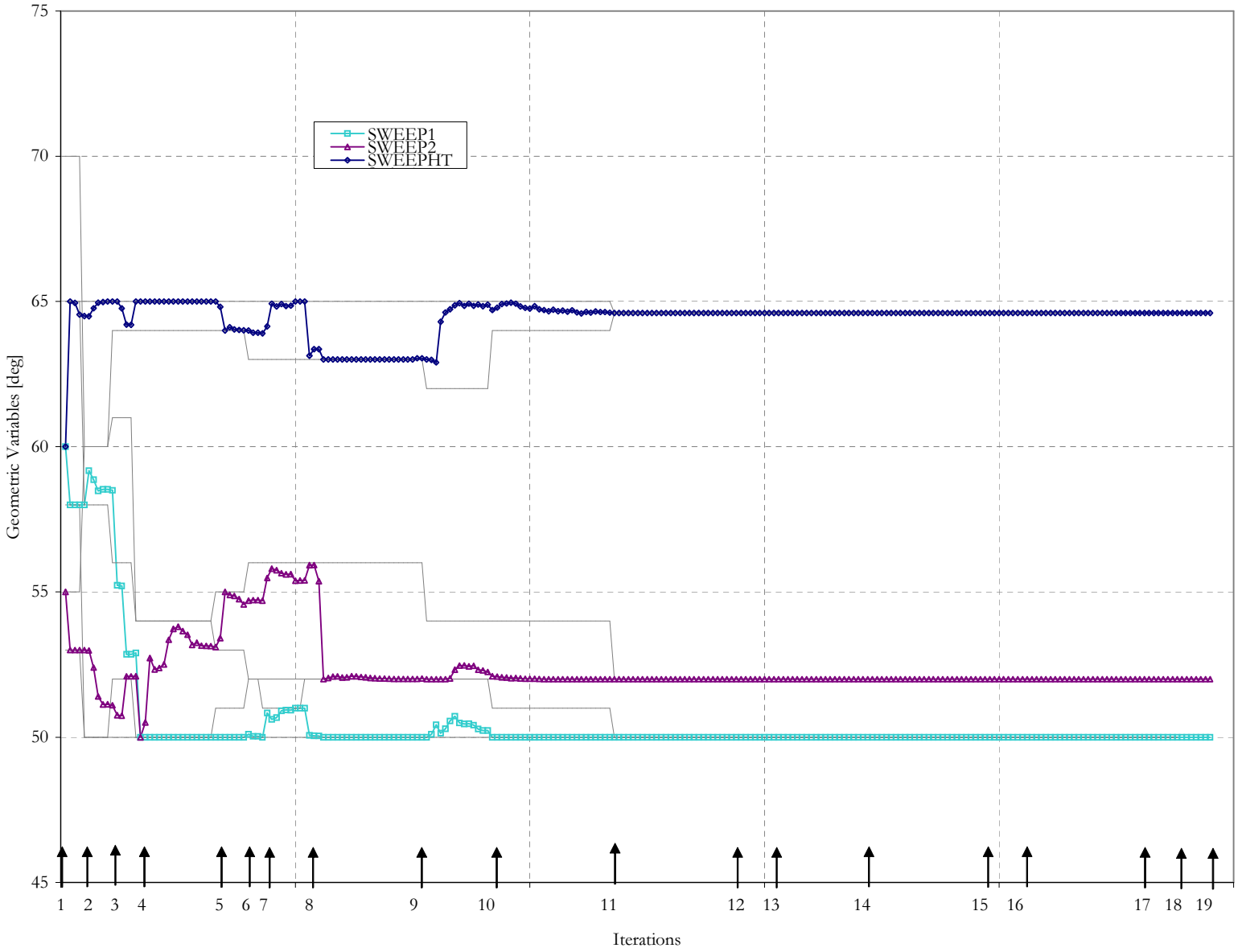

Figure 154: Subsonic Vehicle - Geometric Variables, Sweep

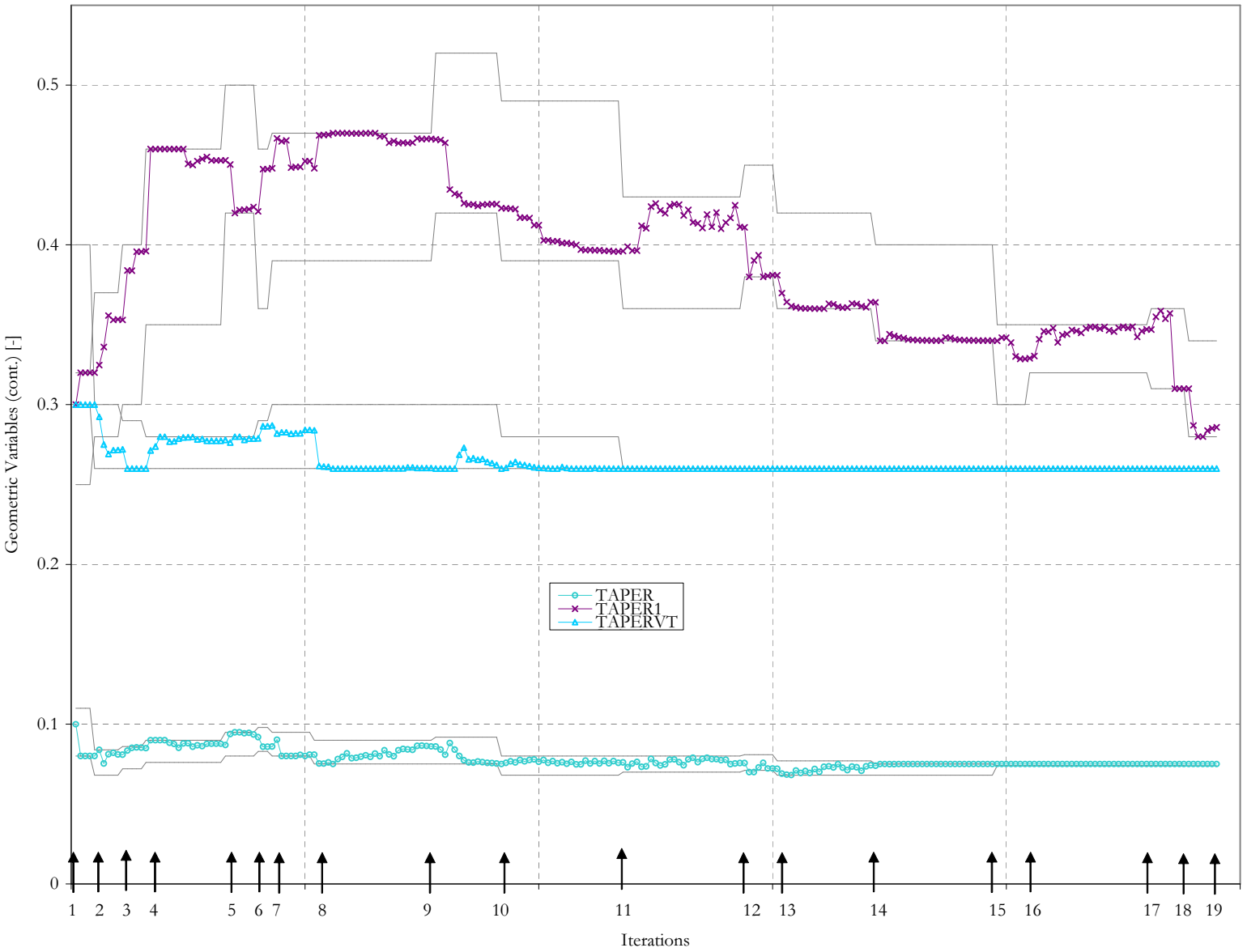

Figure 155: Subsonic Vehicle - Geometric Variables, Taper

D.6 High-Fidelity Aerodynamics Optimization Run

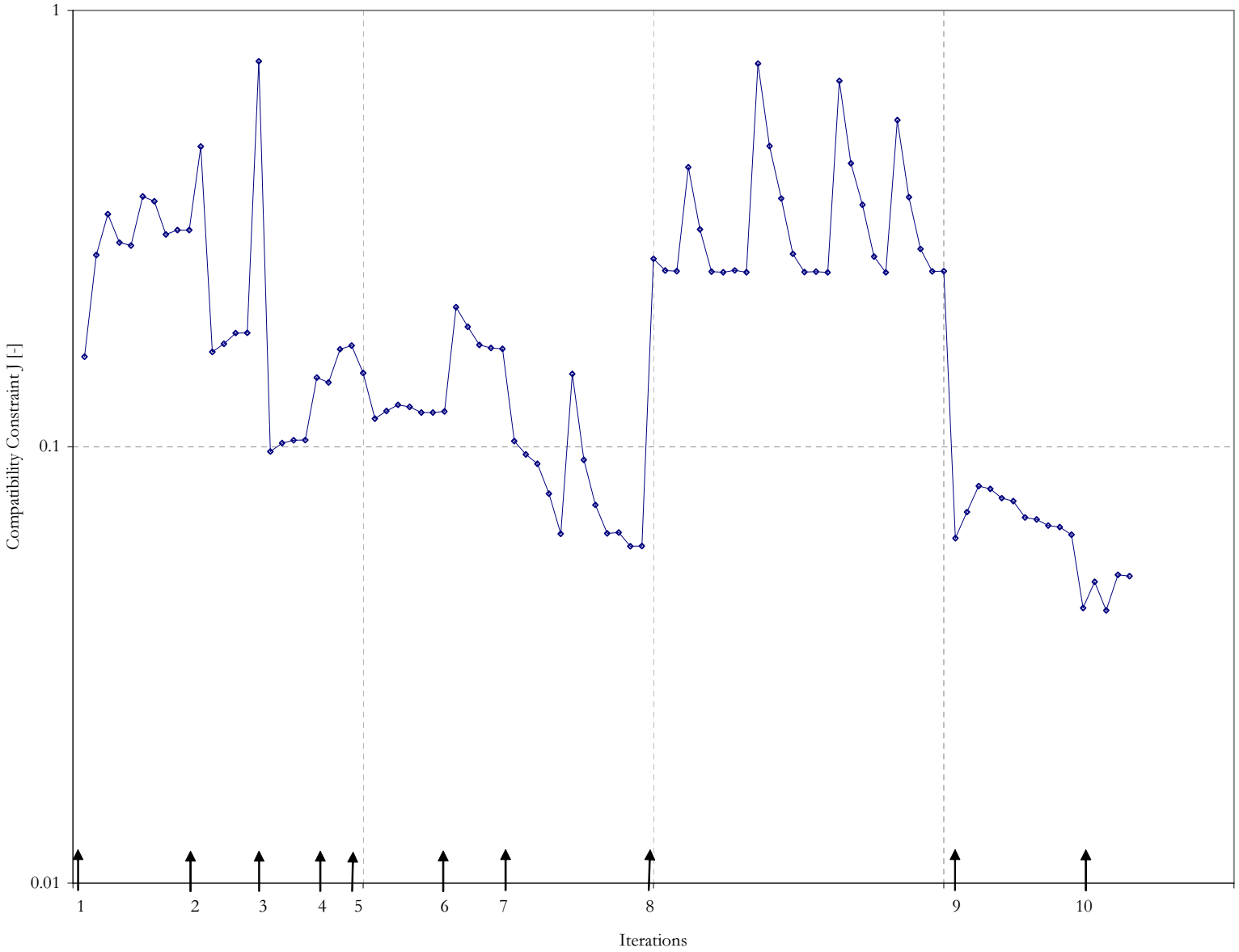

Figure 156: High-Fidelity Aerodynamics Vehicle - Compatibility Constraint J

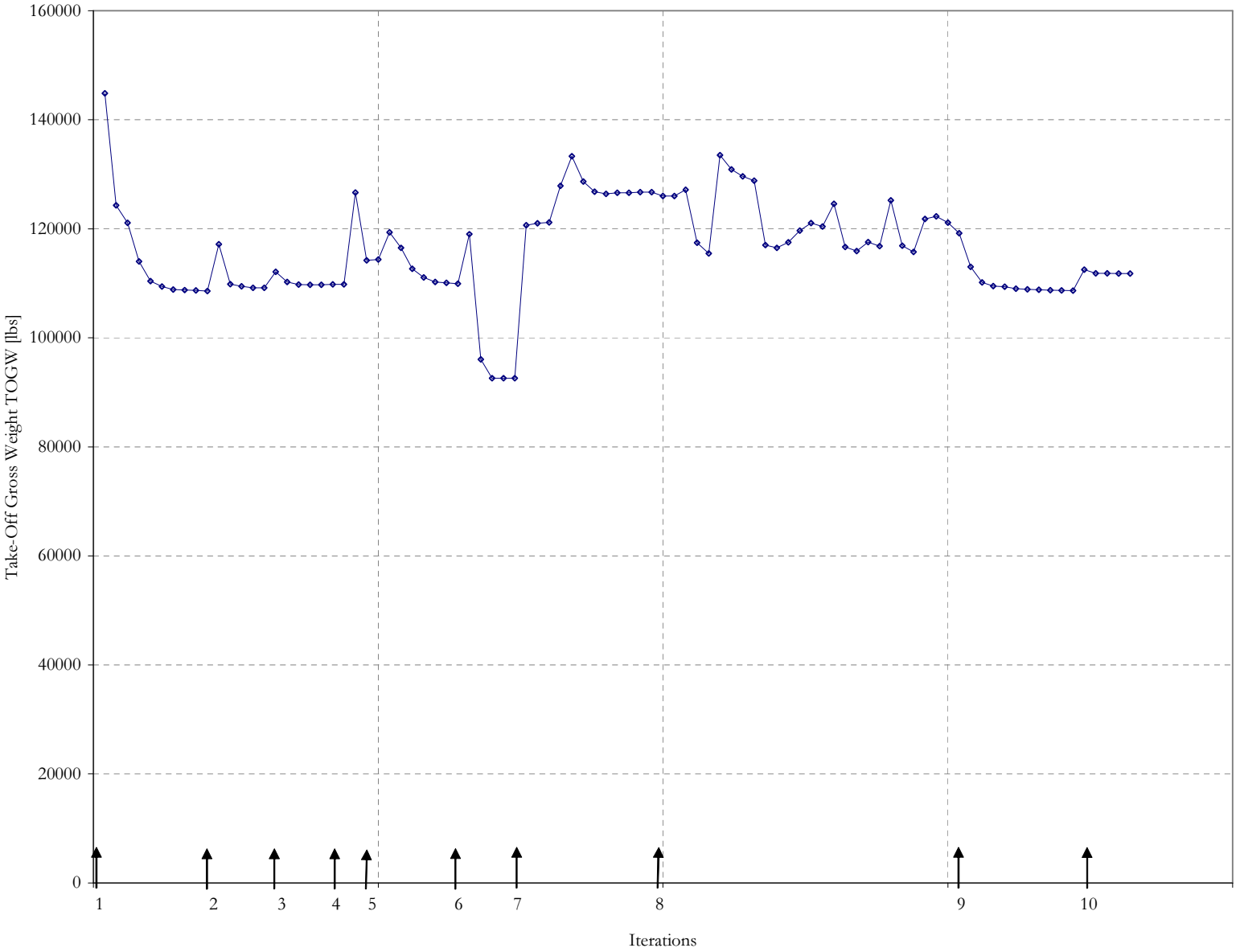

Figure 157: High-Fidelity Aerodynamics Vehicle - Take-off Gross Weight

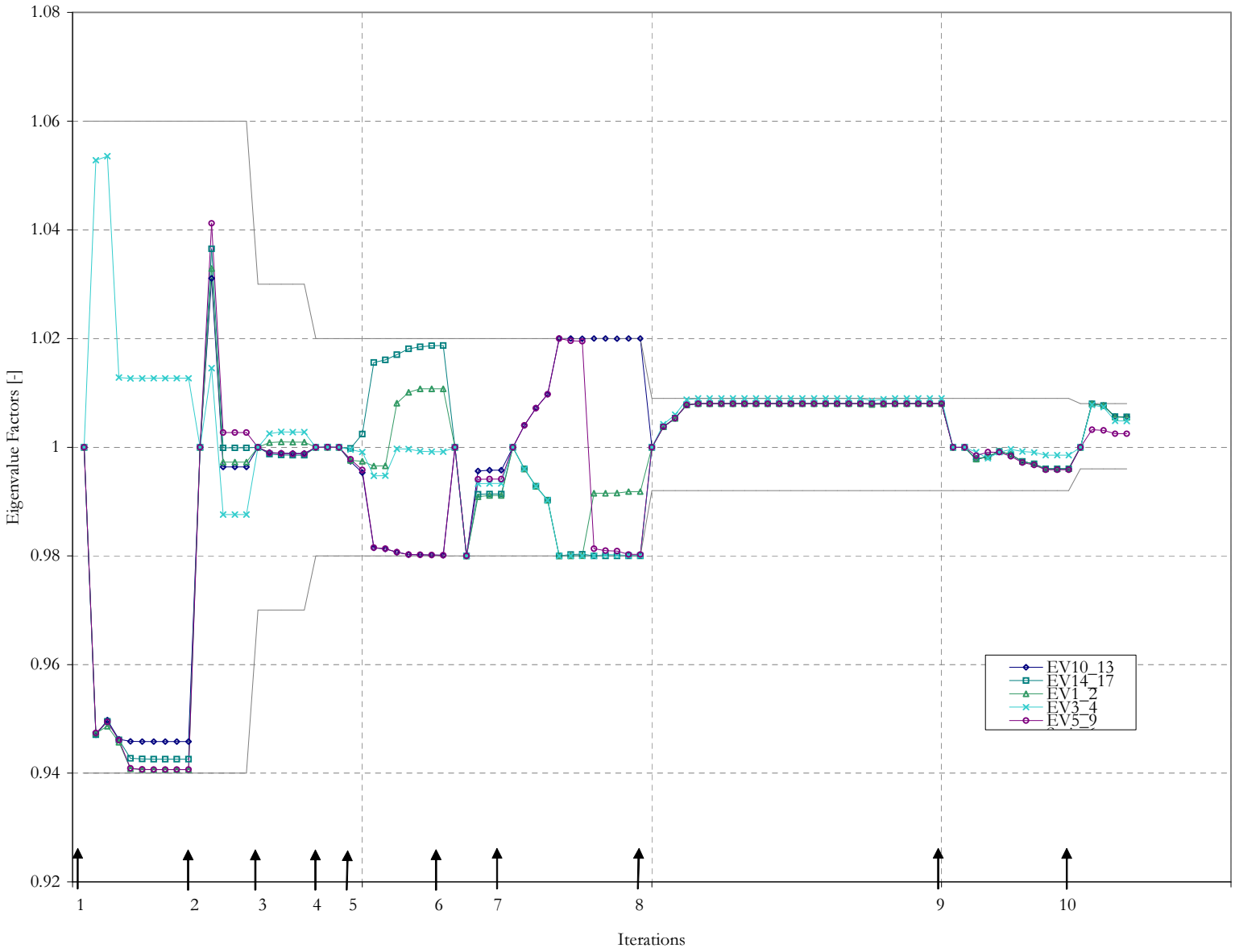

Figure 158: High-Fidelity Aerodynamics Vehicle - Eigenvalue Pooling Factors

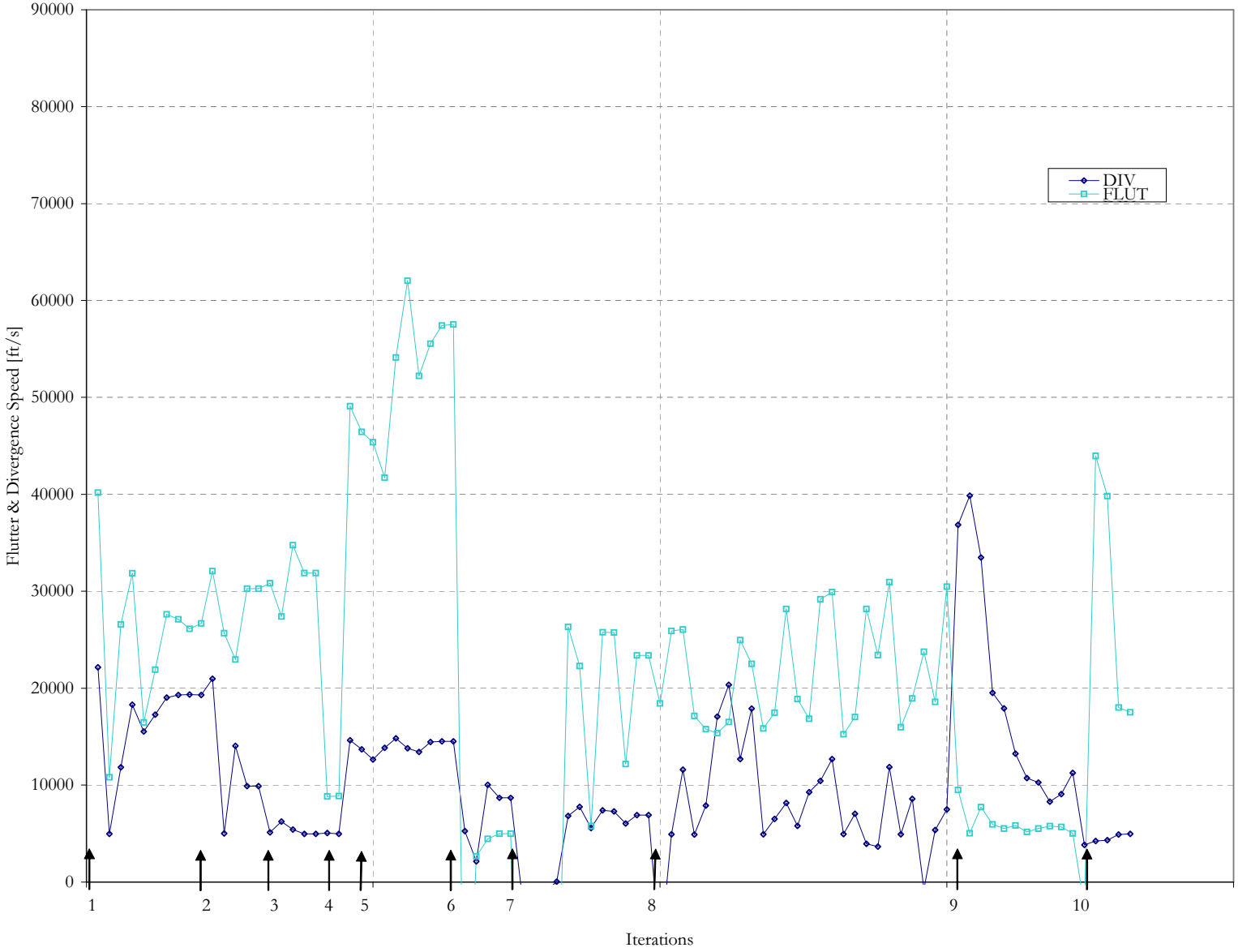

Figure 159: High-Fidelity Aerodynamics Vehicle - Flutter and Divergence Speed

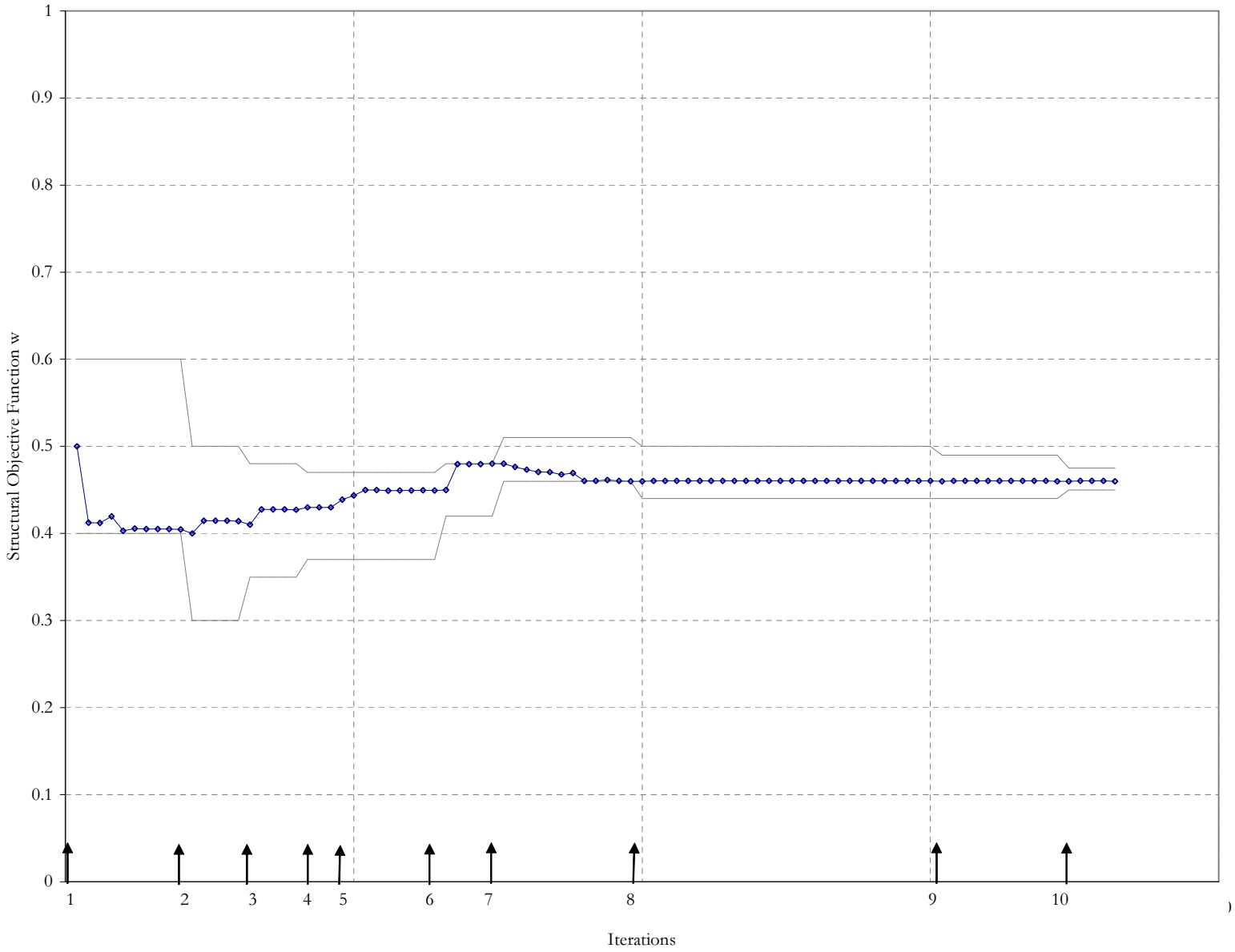

Figure 160: High-Fidelity Aerodynamics Vehicle - Structural Objective Weight w

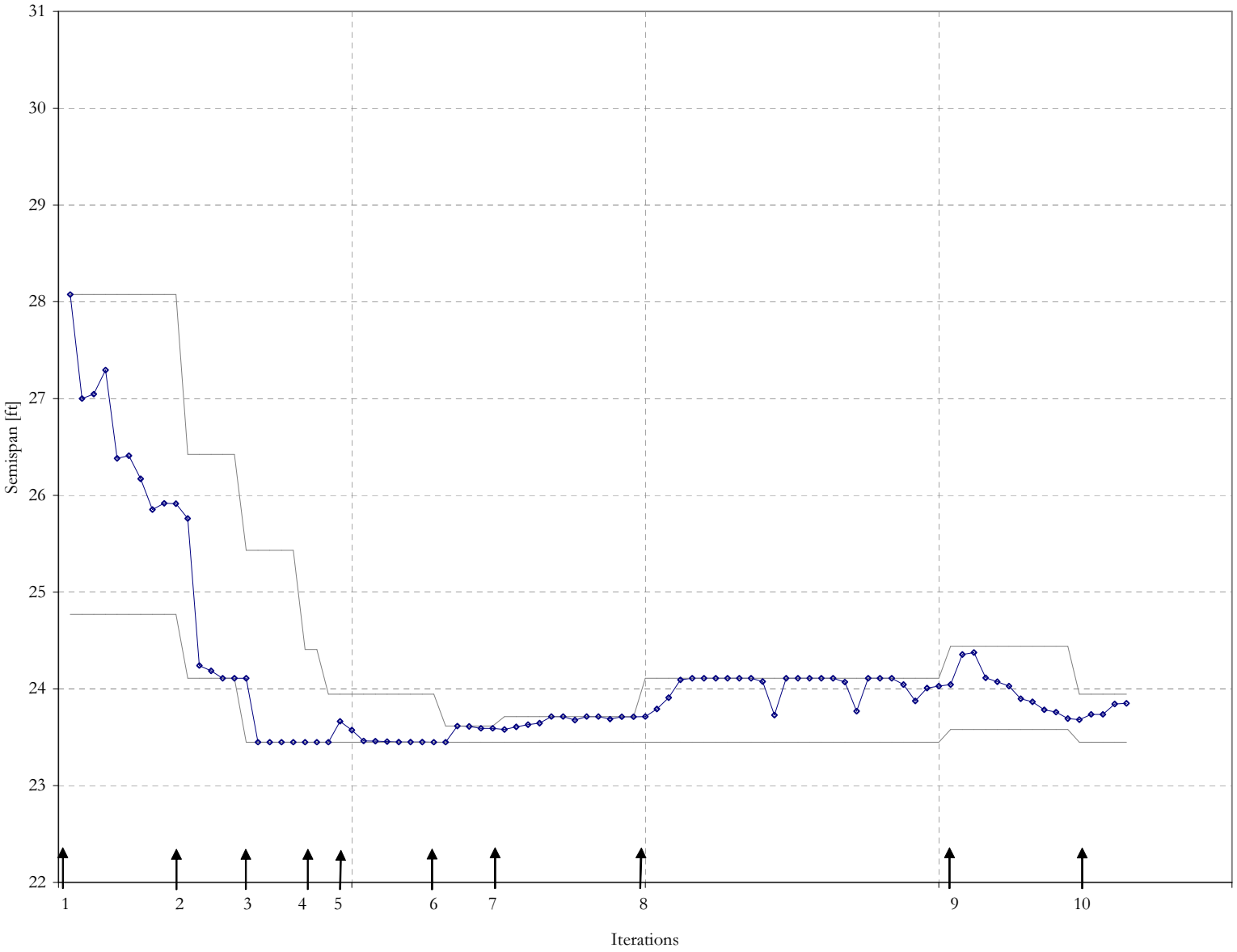

Figure 161: High-Fidelity Aerodynamics Vehicle - Geometric Variables, Semi-span

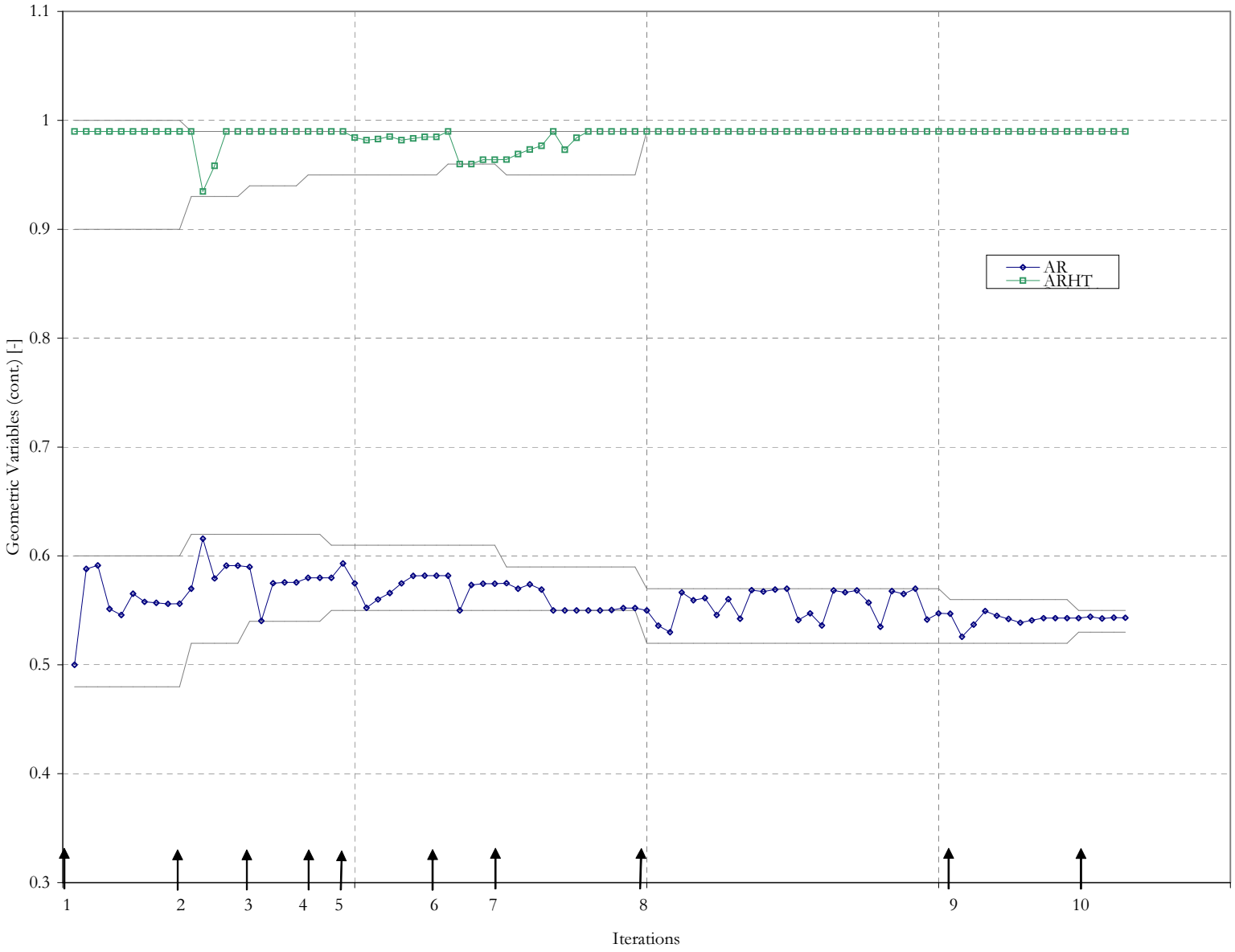

Figure 162: High-Fidelity Aerodynamics Vehicle - Geometric Variables, Aspect Ratio

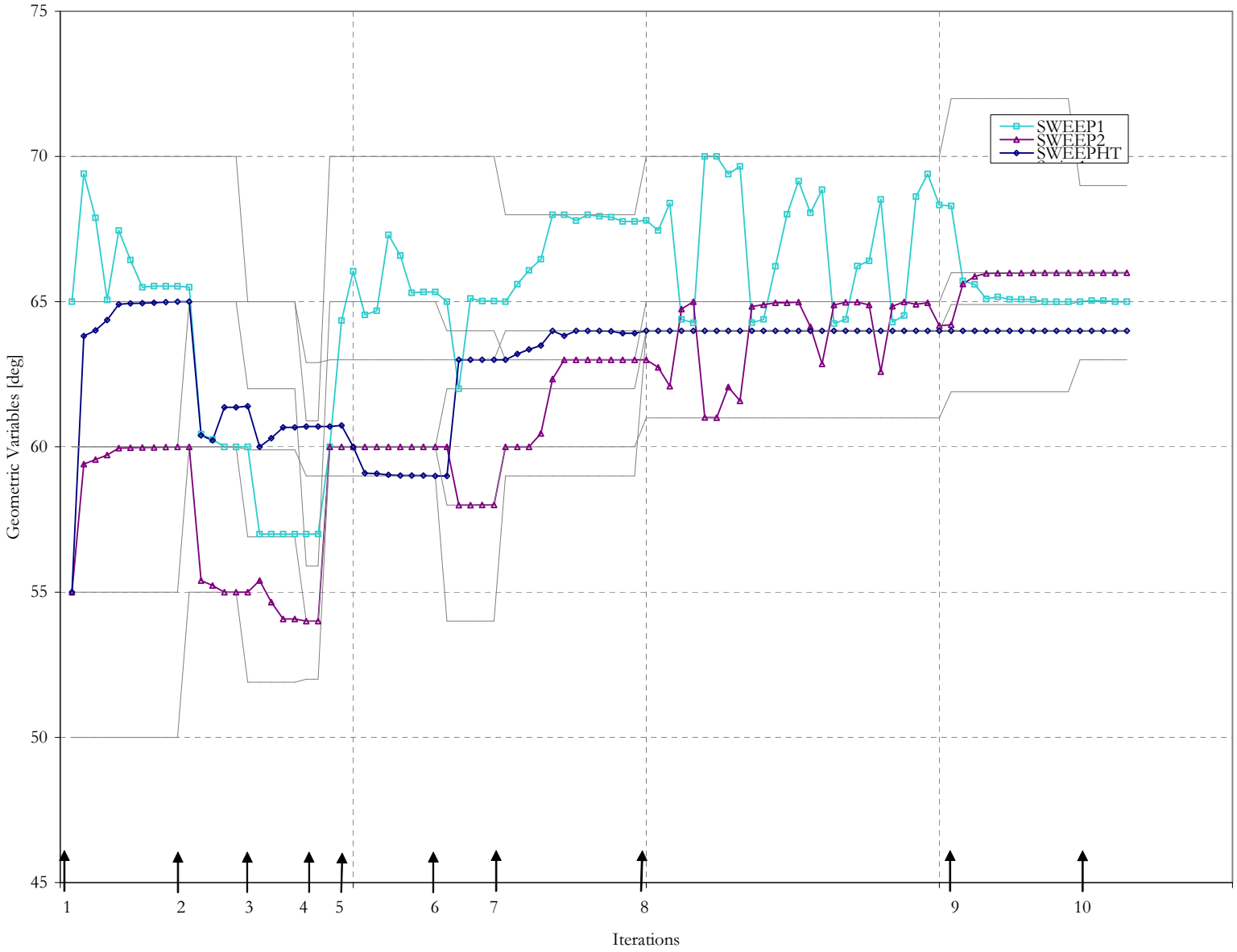

Figure 163: High-Fidelity Aerodynamics Vehicle - Geometric Variables, Sweep

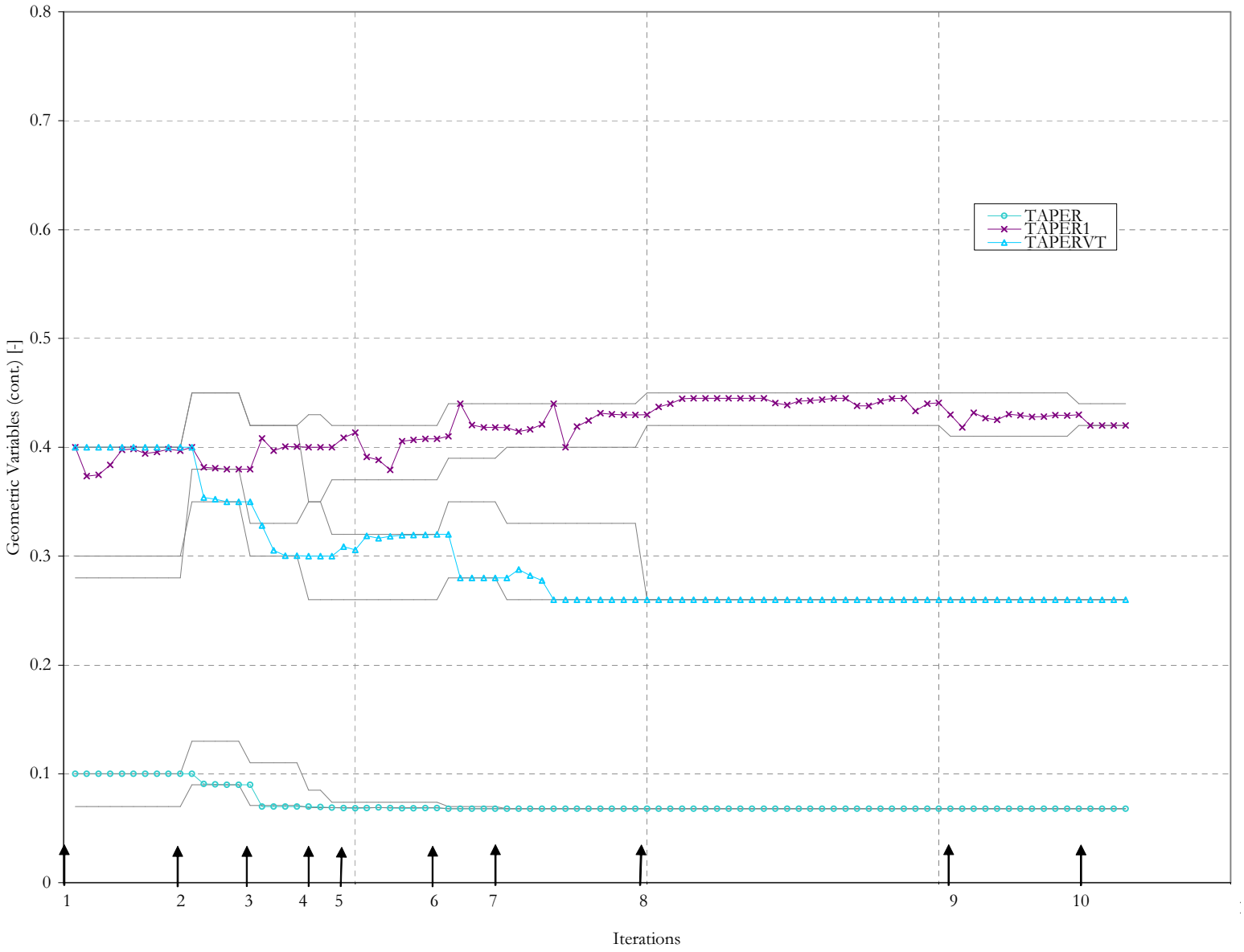

Figure 164: High-Fidelity Aerodynamics Vehicle - Geometric Variables, Taper

REFERENCES

[1] "ACSYNT Overview and Installation Manual," tech. rep., ACSYNT Institute, May 1992.

[2] "The Omega Project for Statistical Computing." `http://www.omegahat.org/`, May 2002.

[3] "The R Project for Statistical Computing." `http://www.r-project.org/`, May 2002.

[4] AIAA Technical Committee on Multidisciplinary Design Optimization, "White Paper on the State of the Art," tech. rep., AIAA, Washington, D.C., Janaury 1991.

[5] Airframe Technology Industry Group, "Aero/Structures/Controls Interaction Working Group - Final Report," tech. rep., Ohio Aerospace Institute, November 1997.

[6] Albright, A. E., Charles, C. J., and Hegedus, M. C., "Modification and Validation of Conceptual Design Aerodynamic Prediction Method HASC95 with VTXCHN," Tech. Rep. NASA-CR-4712, NASSA, March 1996.

[7] Alexander, Jr., M. M., "Structural Problems Associated with Variable Geometry," in *AIAA/RAeS/JSASS Aircraft Design and Technology Meeting*, no. AIAA 65-774, AIAA, November 1965.

[8] ANSYS, Inc., *ANSYS 7.1 Documentation*. ANSYS, Inc., 2003.

[9] Ashley, H., "The Constructive Uses of Aeroelasticity," *Polish Academy of Sciences, Engineering Transactions*, vol. 30, no. 3-4, pp. 369–396, 1982.

[10] B. Hibbard, "VisAD - Homepage." `http://www.ssec.wisc.edu/~billh/`, May 2002.

[11] Battoo, R. S., "Aeroelasticity DAeT 9227." Cranfield University Notes, October 1998.

[12] Battoo, R. S., "An Introductory Guide to Literature in Aeroelasticity," *The Aeronautical Journal*, pp. 511–518, 1999.

[13] Bauchau, O., "Aeroelasticity AE 6200." Georgia Institute of Technology Notes, January 2000.

[14] Bendiksen, O. O., *Experimental Aeroelasticity in Wind Tunnels - History, Status, and Future in Brief*, vol. 5, ch. 5, part 2, p. 243. ASME, 1992.

[15] Bennett, R. M. and Edwards, J. W., "An Overview of Recent Developments in Computational Aeroelasticity," in *29th AIAA Fluid Dynamics Conference*, no. 98-2421, June 1998.

[16] Bisplinghoff, R. L., Ashley, H., and Halfman, R. L., *Aeroelasticity*. Dover, 1996.

[17] Box, E. P. and Draper, N. R., *Empirical Model-Building and Response Surfaces*. John Wiley & Sons, 1987.

[18] Braun, R., Gage, P., Kroo, I., and Sobieski, I., "Implementation and Performance Issues in Collaborative Optimization," in *6th Annual AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, AIAA, AIAA, September 1996.

[19] Buonanno, M. A., "GSRP Renewal, Quantitative Method of Concept Down-Selection from Large Combinatorial Design Spaces," tech. rep., NASA, 2004.

[20] Chen, P. C., Sarhaddi, D., and Liu, D. D., "Development of the Aerodynamic/Aeroservoelastic Modules in ASTROS, Volume 4: ZAERO Theoretical Manual," Tech. Rep. AFRL-VA-WP-TR-1999-3052, Air Froce Research Laboratory, February 1999.

[21] Coen, P., "Development of a Computer Technique for the Prediction of Transport Aircraft Flight Profile Sonic Boom Signatures," Master's thesis, The George Washington University, 1991.

[22] Collar, A. R., "The First Fifty Years of Aeroelasticity," *Aerospace*, February 1978.

[23] DeLaurentis, D. A., *A Probabilistic Approach To Aircraft Design Emphasizing Stability and Control Uncertainties*. PhD thesis, Georgia Institute of Technology, November 1998.

[24] DeLaurentis, D. A., Zink, P. S., Mavris, D. N., Cesnik, C. E., and Schrage, D. P., "New Approaches to Multidisciplinary Synthesis: An Aero-Structures-Control Application Using Statistical Techniques," in *SAE and AIAA, World Aviation Congress, 1st*, no. 965501, October 1996.

[25] Dodd, A. J., Kadrinka, K. E., Loikkanen, M. J., Rommel, B. A., Sikes, G. D., Strong, R. C., and Tzong, T. J., "Aeroelastic Design Optimization Program," *Journal of Aircraft*, vol. 27, pp. 1028–1036, December 1999.

[26] Dowell, E. H., "Eigenmode Analysis in Unsteady Aerodynamics: Reduced Order Models," *AIAA Journal*, vol. 34, pp. 1578–1583, August 1996.

[27] D'Vari, R. and Baker, M., "Aeroelastic Loads and Sensitivity Analysis for Structural Loads Optimization," *Journal of Aircraft*, vol. 36, pp. 156–166, January-February 1999.

[28] Edwards, J. W. and et al, "Transonic Shock Oscillations and Wing Flutter Calculated with an Interactive Boundary Layer Coupling Method," Tech. Rep. NASA-TM-110484, NASA, 1996.

[29] Elliott, D. W., Hoskins, P. D., and Miller, R. F., "A Variable Geometry HSCT," in *AIAA Aircraft Design Systems and Operations Meeting*, no. AIAA 91-3101, AIAA, September 1991.

[30] Engineous Software, Inc., "Engineous Software - Homepage." `http://www.engineous.com/`, August 2002.

[31] Fung, Y. C., *An Introduction to the Theory of Aeroelasticity*. Dover Publications, Inc., 1993 ed., 1969.

[32] Galloway, T., Gelhausen, P., Moore, M., and Waters, M., "Oblique Wing Supersonic Transport Concepts," in *Aircraft Design Systems Meeting*, no. 92-4230, p. 11, AIAA, August 1992.

[33] Georghiades, G. A., Guo, S. J., and Banerjee, J. R., "Flutter Characteristics of Laminated Composite Wings," *Journal of Aircraft*, vol. 33, no. 6, pp. 1204–1206, 1996.

[34] Giles, G. L., "Equivalent Plate Analysis of Aircraft Wing Box Structures with General Planform Geometry," *Journal of Aircraft*, vol. 23, no. 11, pp. 859–864, 1986.

[35] Giunta, A. A., "Sensitivity Analysis Method for Aeroelastic Aircraft Models," *Aircraft Design*, vol. 2, pp. 207–230, December 1999.

[36] Goodwill, J., *Apache Jakarta-Tomcat*. Apress, 2001.

[37] Greene, W. and Sobieszczanski-Sobieski, J., "Minimum Mass Sizing of a large Low Aspect Ratio Airframe for Flutter Free Performance," Tech. Rep. 228-234, Journal of Aircraft, March, 1982 19.

[38] Grossman, B., Haftka, R. T., Mason, W. H., Watson, L. T., Baker, C., Balabanov, V., Cox, S., Giunta, A., Kim, H., D, K., and Krasteva, D., "Effective Use of Surrogate Models in Aircraft Design." `http://www.imm.dtu.dk/~km/smsmeo/contributed/grossman-surrogate.pdf`, February 2004.

[39] Hale, M. A. and Craig, J. I., "IMAGE: A Design Integration Framework Applied to the High Speed Civil Transport," in *1st Industry/Acedemy Symposium on Research for Future Supersonic and Hypersonic Vehicles*, no. HM301, December 1994.

[40] Hale, M. A. and Craig, J. I., "Techniques for Integrating Computer Programs into Design Architectures," in *6th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, no. AIAA-96-4166, September 1996.

[41] Hale, M. A. and Mavris, D. N., "A Lean-Server Foundation for Collaborative Design," in *5th NASA National Symposium on Large-Scale Analysis, Design and Intelligent Synthesis Environments*, October 1999.

[42] Harold, E. R. and Means, W. S., *XML in a Nutshell*. O'Reilly, first ed., January 2001.

[43] Hayter, A. J., *Probability and Statistics for Engineers and Scientists*. PWS Publsihing Company, 1996.

[44] Jimeno, J., "A Physics Based Robust Methodology for Aerodynamic Design Analysis and Optimization," Master's thesis, Georgia Institute of Technology, June 2000.

[45] Johnson, E. H. and Venkayya, V. B., "Automated Structural Optimization System (ASTROS), Volume 1 - Theoretical Manual," Tech. Rep. AFWAL-TR-3028, Air Force Wright Laboratory, December 1988.

[46] Jones, R. T., "Oblique-Wing Supersonic Aircraft," Tech. Rep. 3,971,535, United States Patent and Trademark Office, July 1976.

[47] Jones, R. T. and Nisbet, J. W., " Transonic Transport Wings - Oblique or Swept," *Astronautics and Aeronautics*, vol. 12, pp. 40–47, January 1974.

[48] Karpel, M., "Modal-Based Structural Optimization Using ASTROS," Tech. Rep. T.A.E. No. 795, Technion, September 1997.

[49] Karpel, M., "Reduced-Order Models for Integrated Aeroservoelastic Optimization," *Journal of Aircaft*, vol. 36, pp. 146–155, January-February 1999.

[50] Karpel, M. and Idan, M., "Aeroservoelastic Discipline in ASTROS - Theoretical Manual," Tech. Rep. T.A.E. 820, Technion, January 1999.

[51] Karpel, M. and Moulin, B., "Aeroservoelastic Discipline in ASTROS - User's Manual," Tech. Rep. T.A.E. 834, Technion, January 1999.

[52] Karpel, M., Moulin, B., Anguita, L., Manderuelo, C., and Climent, H., "Flutter Analysis of Aircraft with External Stores Using Modal Coupling," in *43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, no. 2002-1597, April 2002.

[53] Knill, D. L., Giunta, A. A., Baker, C. A., Grossman, B., Mason, W. H., Haftka, R. T., and Watson, L. T., "Response Surface Models Combining Linear and Euler Aerodynamics for Superosnic Transport Design," *Journal of Aircaft*, vol. 36, pp. 75–86, January-February 1999.

[54] Knoernschild, K., *Java Design: Objects, UML, and Process*. Addison-Wesley, 2001.

[55] Koch, P. N., *Hierarchical Modeling and Robust Sythesis for the Preliminary Design of Large Scale Complex Systems*. PhD thesis, Georgia Institute of Technology, December 1997.

[56] Kodiyalam, S. and Yuan, C., "Evaluation of Methods for Multidisciplinary Design Optimization (MDO), Part II," tech. rep., NASA Langley Research Center, November 2000.

[57] Komarov, V. A. and Weisshaar, T. A., "Aircraft Structural Design - Improving Conceptual Design Level Fidelity," in *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, no. 98-4885, AIAA, September 1998.

[58] Kroo, I., "The Aerodynamic Design of Oblique Wing Aircraft," in *AIAA/AHS/ASEE Aircraft Systems Design and Technology Meeting*, no. AIAA 86-2624, AIAA, October 1986.

[59] Largent, M. and Rossow, J., "Structures Design Team." Aerospace Systems Design Laboratory, Georgia Institute of Technology, May 1998.

[60] Lebourg, S., "HISAC - High Speed Aircraft." `http://www.aerosme.com/download/WorkshopsFP6/docs/HISAC.pdf`, October 2002.

[61] Leonard, T. and Hsu, J. S. J., *Bayesian Methods - An Analysis for Statisticians and Interdisciplinary Researchers*. Cambridge University Press, 2001 ed., 1999.

[62] Lillico, M., Butler, R., and Holden, M., " Conceptual Design Optimization of Composite Wings With Aeroelastic and Strength Constraints," in *6th AIAA, NASA, and ISSMO, Symposium on Multidisciplinary Analysis and Optimization*, no. A96-38701, pp. 191–200, AIAA, September 1996.

[63] Loader, C., *Local Regression and Likelihood*. Springer-Verlag, 1999.

[64] MacMillin, P. E., Golovidov, O., Mason, W. H., Grossman, B., and Haftka, R. T., "An MDO Investigation of the Impact of Practical Constraints on an HSCT Configuration," in *35th Aerospace Sciences Meeting and Exhibit*, no. 97-0098, January 1997.

[65] Marx, W. J., Schrage, D. P., and Mavris, D. N., "An Application of Artificial Intelligence for Computer-Aided Design and Manufacturing," in *International Conference on Computational Engineering Science: Supercomputing in Multidisciplinary Analysis and Design*, August 1995.

[66] Mattingly, J. D., Heiser, W. H., and Daley, D. H., *Aircraft Engine Design*. AIAA Education Series, sixth ed., 1987.

[67] Mavris, D. N., Briceno, S. I., Buonanno, M., and Fernandez, I., "A Parametric Exploration of Supersonic Business Jet Concepts Utilizing Response Surfaces," in *2nd AIAA ATIO Forum*, no. 2002-5808, October 2002.

[68] Mavris, D. N. and Hayden, W. T., "Probabilistic Analysis of an HSCT Modeled with an Equivalent Laminated Plate Wing," in *2nd World Aviation Congress*, no. 97-5571, October 1996.

[69] Mavris, D. N., Kamdar, N., Smith, M., Thomas, R., and Wikler, J., "Response Surface Utilization in the Exploration of a Supersonic Business Jet Concept with Application of Emerging Technologies," in *SAE World Aviation Congress*, no. 2003-01-3059, September 2003.

[70] McCullers, L. A., "User's Guide for the Revised Wave Drag Analysis Program," tech. rep., NASA Langley Research Center, 1982.

[71] McCullers, L. A., "Flight Optimization System, User's Guide Version 5.95," tech. rep., NASA Langley Research Center, April 2001.

[72] McDonald, R. A. and Mavris, D. N., "Formulation,Realization, and Demonstration of a Process to Generate Aerodynamic Metamodels for Hypersonic Cruise Vehicle Design," in *5th World Aviation Congress and Exposition*, no. 2000-01-5559, October 2000.

[73] McDonnell Douglas Aerospace - St. Louis (Boeing Aricraft and Missiles), "Finite Element Model Weight Estimation System." `http://www.bmpcoe.org/bestpractices/internal/mdasl/mdasl_27.html`, August 2002.

[74] McLaughlin, B., *Java & XML*. O'Reilly, 2001.

[75] Meridian International Research, "The Market Potential for the SSBJ." `http://www.mirl.demon.co.uk/ssbj.htm/`, December 1999.

[76] Miranda, L. R., Elliot, R. D., and Baker, W. M., "A Generalized Vortex Lattice Method for Subsonic and Supersonic Applications," Tech. Rep. NASA-CR-2865, NASA, December 1977.

[77] Molloy, M. H., "High-Speed Flight and the Military," Tech. Rep. AU/ACSC/136/1999-04, Air Command and Staff College - Air University, April 1999.

[78] Montgomery, D. C., *Design and Analysis of Experiments*. John Wiley & Sons, fifth ed., 2001.

[79] Morris, A. J., "Supporting Distributed Multi-disciplinary Design and Optimization Teams," *Cranfield Univeristy Aerogram*, vol. 10, pp. 21–26, March 2002.

[80] Morris, A. J., Syamsudin, H., Fielding, J. P., Guenov, M., Payne, K. H., Deasley, P. J., Evans, S., and Thorne, J., "MACRO - A tool to support distributed MDO," in *8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, no. 2000-4702, September 2000.

[81] NASA, "Aerodynamic Roadmap from NASA," tech. rep., National Aeronautics and Space Administration, 2002.

[82] Neill, D. J., Johnson, E. H., and Canfield, R., "ASTROS - A Multidisciplinary Automated Structural Design Tool," *Journal of Aircraft*, vol. 27, pp. 1021–1027, December 1990.

[83] Nelms, Jr., W. P., "Applications of Oblique-Wing Technology - An Overview," in *AIAA Aircraft Systems and Technology Meeting*, no. AIAA 76-943, AIAA, September 1976.

[84] Neter, J., Kutner, M. H., Nachtsheim, C. J., and Wasserman, W., *Applied Linear Statistical Models*. Irwin, fourth ed., 1996.

[85] Phoenix Analysis and Design Technologies, Inc., "Phoenix Analysis and Design Technologies Homepage." `http://www.padtinc.com/`, July 2003.

[86] Phoenix Integration, Inc., "Improving The Engineering Process with Software Integration - Integrating Engineering Applications for Design." `www.phoenix-int.com/publications/` , July 2002.

[87] Phoenix Integration, Inc., "Phoenix Integration: Software for Engineers: Using ModelCenter and Analysis Server Worldwide - Homepage." `http://www.phoenix-int.com/`, August 2002.

[88] Raveh, D. E., "Reduced Order Models for Nonlinear Unsteady Aerodynamics," in *7th Annual AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, no. 2000-4785, September 2000.

[89] Raymer, D. P., *Aircraft Design: A Conceptual Approach*. AIAA, 1999.

[90] Ricketts, R. H., *Experimental Aeroelasticity in Wind Tunnels - History, Status, and Future in Brief*, vol. 5, ch. 2, part 2, pp. 151–177. ASME, 1992.

[91] Ricketts, R. H. and J.Sobieszczanski, "Simplified and Refined Structural Modeling for Economical Flutter Analysis and Design," in *8th AIAA/ASME/SAE Structures, Structural Dynamics, and Materials Conference*, no. AIAA-77-421, Mrach 1977.

[92] Riel, A. J., *Object-Oriented Design Heuristics*. Addison-Wesley, 1999.

[93] Ringertz, U. T., "On Structural Optimization with Aeroelastic Constraints," *Structural Optimization*, vol. 8, no. 1, pp. 16–23, 1994.

[94] Rohl, P. J., *A Multilevel Decomposition Procedure for the Preliminary Wing Design of a High-Speed Civil Transport Aircraft*. PhD thesis, Georgia Institute of Technology, May 1995.

[95] Roskam, J., *Airplane Design*. Roskam Aviation and Engineering Corp., 1985.

[96] SAS Institute Inc., "JMP - Statistical Discovery - Data Analysis Software - Six Sigma - Design of Experiments - Statistics Software Homepage." `http://www.jmp.com/`, August 2002.

[97] Scott, A. T. and Olds, J., "An Evaluation of Three Commercially Available Integrated Design Framework Packages for Use in the Space Systems Design Lab," tech. rep., Georgia Institute of Technology, April 2001.

[98] Shirk, M., Hertz, T., and Weishaar, T., "Aeroelastic tailoring: Theory, Practice, and Promise," *Journal of Aircraft*, vol. 23, pp. 6–18, January 1986.

[99] Simpson, T. W., Mauery, T. M., Korte, J. J., and Mistree, F., "Kriging Models for Global Approximation in Simulation-Based Multidisciplinary Design Optimization," *AIAA Journal*, vol. 39, pp. 2233–2241, December 2001.

[100] Sobieszczanski-Sobieski, J., "Sensitivity Analysis and Multidisciplinary Optimization for Aircraft Design: Recent Advances and Results," *Journal of Aircraft*, vol. 27, pp. 993–1001, December 1990.

[101] Sobieszczanski-Sobieski, J., "Multidisciplinary Design Optimization: An Emerging New Engineering Discipline," in *World Congress on Optimal Design of Structural Systems*, Kluwer, August 1993.

[102] Sobieszczanski-Sobieski, J., Agte, J. S., and Sandusky Jr., R. R., "Bi-Level Integrated System Synthesis," *AIAA Journal*, vol. 38, pp. 164–172, January 2000.

[103] Sobieszczanski-Sobieski, J., Altus, T. D., Phillips, M., and Sandusky, R., "Bi-Level Integrated System Synthesis (BLISS) for Concurrent and Distributed Processing," in *9th AIAA/ISSMO Multidisciplinary Analysis and Optimization Symposium*, no. 2002-5409, September 2002.

[104] Sobieszczanski-Sobieski, J. and Haftka, R. T., "Multidsciplinary Aerospace Design Optimization: Survey of Recent Developments," in *34th Aerospace Sciences Meeting and Exhibit*, no. 96-0711, January 1996.

[105] Sobieszczanski-Sobieski, J. and Venter, G., "Imparting Desired Attributes by Optimization in Structural Design," in *44th AIAA/ASME/ASCE/AHS Structures, Structural Dynamics, and Materials Conference*, no. 2003-1546, April 2003.

[106] SpaceWorks Engineering, Inc., "SpaceWorks Engineering, Inc. Homepage." http://www.spaceworks.aero/, October 2003.

[107] Stein, M. L., *Interpolation of Spatial Data: Some Theory for Kriging*. Springer-Verlag, 1999.

[108] Striz, A. G. and Venkayya, V. B., "Influence of Structural and Aerodynamic Modeling on Flutter Analysis," *Journal of Aircraft*, vol. 31, pp. 1205–1211, September-October 1994.

[109] Sun, X. S., Duan, S. H., Sun, X. X., Ding, H. L., Piening, M., and Zimmerman, R., "Aeroelastic Tailoring of Composite Wings by Sequence Quadratic Programming," in *Proceedings of the International Forum on Aeroelasticity and Structural Dynamics*, no. A95-42613 11-39, pp. 791–799, June 1995.

[110] Taylor, R. M. and Weisshaar, T. A., "Structural Information Technologies for Aircraft Design Process Improvement," in *41st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference and Exhibit*, no. 2000-24521, AIAA, April 2000.

[111] TechnoSoft, Inc., "TechnoSoft Inc. - Homepage." http://www.technosoft.com/, August 2002.

[112] The Apache Software Foundation, "Apache SOAP." http://xml.apache.org/soap/, May 2002.

[113] The Apache Software Foundation, "The Apache Software Foundation Homepage." http://www.apache.org/, May 2002.

[114] The MacNeal-Schwendler Corporation, *MSC/NASTRAN Theoretical Manual*. The MacNeal-Schwendler Corporation, 1988.

[115] The MathWorks, Inc., *MATLAB Function Reference*. The MathWorks, Inc., revised for 6.0 ed., 2000.

[116] Thuruthimattam, B. J., Friedmann, P. P., McNamara, J. J., and Powell, K. G., "Aeroelasticity of a Generic Hypersonic Vehicle," in *43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, no. 2002-1209, April 2002.

[117] Torenbeek, E., *Synthesis of Subsonic Airplane Design: An Introduction to the Preliminary Design of Subsonic General Aviation and Transport Aircraft, With Emphasis On Layout, Aerodynamic Design, Propulsion And Performance*. Delft Univeristy Press, 1976.

[118] Vanderplaats Research and Development, Inc., "DOT (Design Optimization Tools) Homepage." http://www.vrand.com/dot.htm, August 2002.

[119] Venkataraman, S. and Haftka, R. T., "Structural Optimization: What Has Moore's Law Done for Us?," in *43rd AIAA/ASME/ASCE/AHS Structures, Structural Dynamics abd Materials Conference*, no. 2002-1342, April 2002.

[120] Wakayama, S. and Kroo, I., "Subsonic Wing Design using Multidisciplinary Optimization," in *5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Pt. 2*, no. 94-4409-CP, pp. 1358–1368, September 1994.

[121] Wakayama, S., Page, M., and Liebeck, R., "Multidisciplinary Optimization On An Advanced Composite Wing," in *6th AIAA/NASA/ISSMO, Symposium on Multidisciplinary Analysis and Optimization*, no. 96-4003-CP, September 1996.

[122] Weisshaar, T. A., *Aeroelasticity*, vol. 5, ch. 1, Pt. 2, pp. 147–150. ASME, 1992.

[123] Welge, H. R., "The Historical Development of a Large Multidisciplinary Technical Program - The High Speed Research Program and Environmental Compatibility Achievements." Presented at Aerospace Congress and Exhibition, September 2003.

[124] Wiler, C. D. and White, S. N., "Projected Advantage of an Oblique Wing Design on a Fighter Mission," in *AIAA, AHS, ASEE, Aircraft Design Systems and Operations Meeting*, no. AIAA 84-2474, AIAA, November 1984.

[125] Wilhite, A. and Shaw, R. J., "NASA's High Speed Research Program - Revolutionary technology leaps to eliminate barriers for affordable supersonic travel," tech. rep., `http://oea.larc.nasa.gov/HSR/aa_article.html`, August 1998.

[126] Willcox, K. E., *Reduced-Order Aerodynamic Models for Aeroelastic Control of Turbomachines*. PhD thesis, Massachusetts Institute of Technology, February 2000.

[127] World Wibe Web Consortium, "Extensible Markup Language (XML) 1.0 (Second Edition)." `http://www.w3c.org/TR/REC-xml/`, May 2002.

[128] World Wibe Web Consortium, "Simple Object Access Protocol (SOAP) 1.1." `http://www.w3c.org/TR/SOAP/`, May 2002.

[129] Wright, B. R., Bruckman, F., and Radovcich, N. A., "Arrow Wings for Supersonic Cruise Aircraft," in *AIAA 16th Aerospace Sciences Meeting*, no. AIAA 78-151, AIAA, January 1978.

[130] Young, J. A., Anderson, R. D., and Yurkovich, R. N., "A Description of the F/A-18E/F Design and Design Process," in *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, no. 98-4701, September 1998.

[131] Yuan, C., "Multidisciplinary Design Optimization." http://www.sightnacorp.com/download/501.MDO.pdf, September 2003.

[132] Yurkovich, R., "Application of the ASTROS Code in an Advanced Design Environment." Presented at ASTROS Workshop, September 1994.

[133] Zink, P. S., *A Methodology for Robust Structural Designs with Application to Active Aeroelastic Wings*. PhD thesis, Georgia Institute of Technology, June 2001.

[134] Zink, P. S., DeLaurentis, D. A., Hale, M. A., Volovio, V. V., Schrage, D. P., Craig, J. I., Fulton, R. E., Mistree, F., and Mavris, D. N., "New Approaches To High Speed Civil Transport Multidisciplinary Design and Optimization," tech. rep., IEEE, 2000.

[135] Zona Technology, Inc., "ZAERO Version 6.2 Theoretical Manual," Tech. Rep. Seventeenth Edition, Zona Technology, Inc., October 2002.

VITA

Peter De Baets was born in Roeselare, Belgium on October 12th, 1976. He completed secondary school at the Klein Seminarie Roeselare (KSRO) in 1994. He then enrolled in the Katholieke Hogeschool Brugge-Oostende (KHBO) in Oostende, Belgium. He received an electro-mechanical engineering degree (Industrieel Ingenieur Elektro-Mechanica) in 1998 from that institution. Since then he earned a Master of Science (Aeronautical Vehicle Design) from Cranfield University, United Kingdom in 1999.